FTGL

2.4.0

Generated by Doxygen 1.9.7

Chapter 1

FTGL User Guide

1.1 Introduction

OpenGL doesn't provide direct font support, so the application must use any of OpenGL's other features for font rendering, such as drawing bitmaps or pixmaps, creating texture maps containing an entire character set, drawing character outlines, or creating a 3D geometry for each character.

More information can be found on the OpenGL website:

- http://www.opengl.org/resources/faq/technical/fonts.htm
- http://www.opengl.org/resources/features/fontsurvey/

Most of these systems require a pre-processing stage to take the native fonts and convert them into a proprietary format.

FTGL was born out of the need to treat fonts in OpenGL applications just like any other application. For example when using Adobe Photoshop or Microsoft Word you don't need an intermediate pre-processing step to use high quality scalable fonts.

1.2 Documentation

- FTGL tutorial
- C API reference:
  - FTGlyph.h
  - FTFont.h
  - FTLayout.h
- C++ API reference:
  - class FTGlyph
  - class FTFont
  - class FTLayout

1.3 Additional information

- Frequently Asked Questions
- Projects using FTGL

1.4 FTGL tutorial

1.4.1 Starting to use %FTGL

Only one header is required to use FTGL:

```
#include <FTGL/ftgl.h>
```

1.4.2 Choosing a font type

FTGL supports 6 font output types among 3 groups: raster fonts, vector fonts, and texture fonts which are a mixture of both. Each font type has its advantages and disadvantages.

1.4.2.1 Raster fonts

Raster fonts are made of pixels painted directly on the viewport's framebuffer. They cannot be directly rotated or scaled.

- Bitmap fonts use 1-bit (2-colour) rasterised glyphs.
- Pixmap fonts use 8-bit (256 levels) rasterised glyphs.

1.4.2.2 Vector fonts

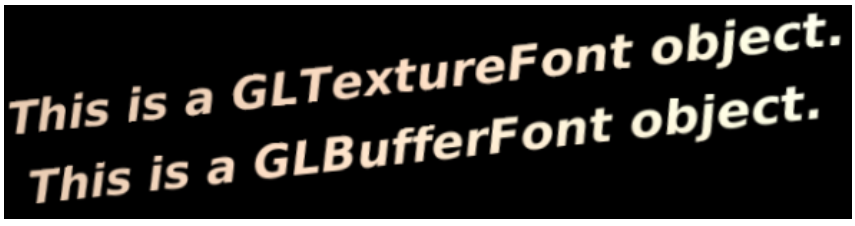
Vector fonts are 3D objects that are rendered at the current matrix location. All position, scale, texture and material effects apply to vector fonts.

- Polygon fonts use planar triangle meshes and can be texture-mapped.
- Outline fonts use OpenGL lines.
- Extruded fonts are extruded polygon fonts, with the front, back and side meshes renderable separately to apply different effects and materials.

1.4.2.3 Textured fonts

Textured fonts are probably the most versatile types. They are fast, antialiased, and can be transformed just like any OpenGL primitive.

- Texture fonts use one texture per glyph. They are fast because glyphs are stored permanently in the video card's memory.
- Buffer fonts use one texture per line of text. They tend to be faster than texture fonts when the same line of text needs to be rendered for more than one frame.



1.4.3 Create font objects

Creating a font and displaying some text is really straightforward, be it in C or in C++.

1.4.3.1 in C

```
/* Create a pixmap font from a TrueType file. */
FTGLfont *font = ftglCreatePixmapFont("/home/user/Arial.ttf");

/* If something went wrong, bail out. */
if(!font)
    return -1;

/* Set the font size and render a small text. */
ftglSetFontFaceSize(font, 72, 72);
ftglRenderFont(font, "Hello World!", FTGL_RENDER_ALL);

/* Destroy the font object. */
ftglDestroyFont(font);
```

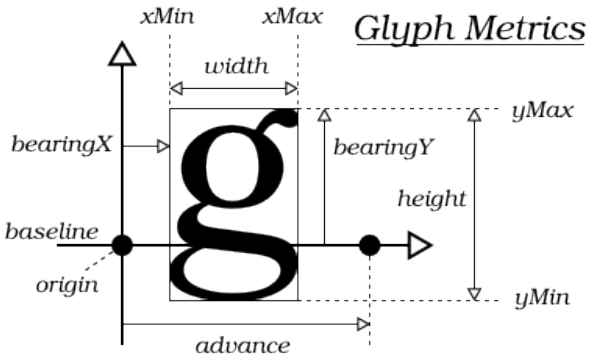
1.4.3.2 in C++

```
// Create a pixmap font from a TrueType file.
FTGLPixmapFont font("/home/user/Arial.ttf");

// If something went wrong, bail out.
if(font.Error())
    return -1;

// Set the font size and render a small text.
font.FaceSize(72);
font.Render("Hello World!");
```

The first 128 glyphs of the font (generally corresponding to the ASCII set) are preloaded. This means that usual text is rendered fast enough, but no memory is wasted loading glyphs that will not be used.

1.4.4 More font commands

1.4.4.1 Font metrics



If you ask a font to render at 0.0, 0.0 the bottom left most pixel or polygon may not be aligned to 0.0, 0.0. With FTFont::Ascender(), FTFont::Descender() and FTFont::Advance() an approximate bounding box can be calculated.

For an exact bounding box, use the FTFont::BBox() function. This function returns the extent of the volume containing 'string'. 0.0 on the y axis will be aligned with the font baseline.

1.4.4.2 Specifying a character map encoding

From the FreeType documentation:

¨By default, when a new face object is created, (FreeType) lists all the charmaps contained in the font face and selects the one that supports Unicode character codes if it finds one. Otherwise, it tries to find support for Latin-1, then ASCII.¨

It then gives up. In this case FTGL will set the charmap to the first it finds in the fonts charmap list. You can expilcitly set the char encoding with FTFont::CharMap().

Valid encodings as of FreeType 2.0.4 are:

- ft_encoding_none
- ft_encoding_unicode
- ft_encoding_symbol
- ft_encoding_latin_1
- ft_encoding_latin_2
- ft_encoding_sjis
- ft_encoding_gb2312
- ft_encoding_big5
- ft_encoding_wansung
- ft_encoding_johab
- ft_encoding_adobe_standard
- ft_encoding_adobe_expert
- ft_encoding_adobe_custom
- ft_encoding_apple_roman

For instance:

```
font.CharMap(ft_encoding_apple_roman);
```

This will return an error if the requested encoding can't be found in the font.

If your application uses Latin-1 characters, you can preload this character set using the following code:

```
// Create a pixmap font from a TrueType file.
FTGLPixmapFont font("/home/user/Arial.ttf");

// If something went wrong, bail out.
if(font.Error())
    return -1;

// Set the face size and the character map. If something went wrong, bail out.
font.FaceSize(72);
if(!font.CharMap(ft_encoding_latin_1))
    return -1;

// Create a string containing all characters between 128 and 255
// and preload the Latin-1 chars without rendering them.
char buf[129];
for(int i = 128; i < 256; i++)
{
    buf[i] = (char)(unsigned char)i;
}
buf[128] = '\0';

font.Advance(buf);
}
```

1.4.5 Sample font manager class

```
FTTextureFont* myFont = FTGLFontManager::Instance().GetFont("arial.ttf", 72);
```

```
#include <map>
#include <string>
#include <FTGL/ftgl.h>

using namespace std;

typedef map<string, FTFont*> FontList;
typedef FontList::const_iterator FontIter;

class FTGLFontManager
{
    public:
        // NOTE
        // This is shown here for brevity. The implementation should be in the source
        // file otherwise your compiler may inline the function resulting in
        // multiple instances of FTGLFontManager
        static FTGLFontManager& Instance()
        {
            static FTGLFontManager tm;
            return tm;
        }

        ~FTGLFontManager()
        {
            FontIter font;
            for(font = fonts.begin(); font != fonts.end(); font++)
            {
                delete (*font).second;
            }

            fonts.clear();
        }


        FTFont* GetFont(const char *filename, int size)
        {
            char buf[256];
            sprintf(buf, "%s%i", filename, size);
            string fontKey = string(buf);

            FontIter result = fonts.find(fontKey);
            if(result != fonts.end())
            {
                LOGMSG("Found font %s in list", filename);
                return result->second;
            }

            FTFont* font = new FTTextureFont;

            string fullname = path + string(filename);

            if(!font->Open(fullname.c_str()))
            {
                LOGERROR("Font %s failed to open", fullname.c_str());
                delete font;
                return NULL;
            }

            if(!font->FaceSize(size))
            {
                LOGERROR("Font %s failed to set size %i", filename, size);
                delete font;
                return NULL;
            }

            fonts[fontKey] = font;

            return font;
        }


    private:
        // Hide these 'cause this is a singleton.
        FTGLFontManager(){}
        FTGLFontManager(const FTGLFontManager&){};
        FTGLFontManager& operator = (const FTGLFontManager&){ return *this; };

        // container for fonts
        FontList fonts;
};
```

1.5 Frequently Asked Questions

1.5.1 FAQ

1.5.1.1 When I try to compile %FTGL it complains about a missing file from the include: #include <ft2build.h>

FTGL relies on FreeType 2 for opening and decoding font files. This include is the main include for FreeType. You will need to download Freetype 2 and install it. Then make sure that the FTGL project that you are using points to your FreeType installation.

1.5.1.2 Is it possible to map a font to a ¨unit¨ size? My application relies on the fonts being a certain ¨physical¨ height (in OpenGL coordinate space) rather than a point size in display space. Any thoughts/suggestions?

We can do anything:) It would be easy to allow you to set the size in pixels, though I'm not sure this is what you want. Setting the size to 'OpenGL units' may be a bit harder. What does 1.0 in opengl space mean and how does that relate to point size? For one person it might mean scaling the font up, for someone else it may mean scaling down. Plus bitmaps and pixmaps have a pixel to pixel relationship that you can't change.

Here's some guidelines for vector and texture fonts. Take note that I say 'should' a lot :)

- One point in pixel space maps to 1 unit in OpenGL space, so a glyph that is 18 points high should be 18.0 units high.
- If you set an ortho projection to the window size and draw a glyph it's screen size should be the correct physical size ie a 72 point glyph on a 72dpi screen will be 1 inch high. Also if you set a perspective projection that maps 0.0 in the z axis to screen size you will get the same eg.
  ```
  gluPerspective(90, window_height / 2 , small_number, large_number);
  ```
  So basically it all depends on your projection matrix. Obviously you can use glScale but I understand if you don't want to.

Couple of extra things to note:

- The quality of vector glyphs will not change when you change the size, ie. a really small polygon glyph up close will look exactly the same as a big one from far away. They both contain the same amount of data. This doesn't apply to texture fonts.
- Secondly, there is a bug in the advance/kerning code that will cause ugliness at really small point sizes. This is because the advance and kerning use ints so an advance of 0.4 will become zero. If this is going to be a probelm, I can fix this.

Early on I did a lot of head scratching over the OpenGL unit to font size thing because when I was first integrating FTGL into my engine the fonts weren't the size I was expecting. I was tempted to build in some scaling but I decided doing nothing was the best approach because you can't please everyone. Plus it's 'correct' as it is.

1.6 Projects using FTGL

To add your project to this list, please contact one of the FTGL developers at http://sf.net/projects/ftgl.

Projects are listed in alphabetical order.

1.6.1 %FTGL language bindings

1.6.1.1 %FTGL#

FTGL# (http://www.paskaluk.com/projects.php) is a collection of .NET bindings for FTGL.

1.6.1.2 GlGuiA

GlGuiA (http://sourceforge.net/projects/glguia/) is a set of packages for Ada 2006 that can be used to create Graphical User Interfaces, relaying (almost) only on OpenGl. Hence should be rather platform-independant.

1.6.1.3 %FTGL

Haskell FTGL (http://www.haskell.org/haskellwiki/FTGL) is a Haskell binding interface to FTGL.

1.6.1.4 Ruby %FTGL

Ruby FTGL# (http://rubyforge.org/projects/ruby-ftgl/) is a collection of Ruby bindings for FTGL.

1.6.1.5 PyFTGL

PyFTGL (http://code.google.com/p/pyftgl/) wraps the functionality of FTGL into a Python module so that it can be used in conjunction with PyOpenGL.

1.6.1.6 Tcl3D

Tcl3D (http://www.tcl3d.org) offers the 3D functionality of OpenGL and other 3D libraries (including FTGL) at the Tcl scripting level.

1.6.2 Projects currently using %FTGL

1.6.2.1 Agent World

Agent World (http://code.google.com/p/agentw/) provides tools for simulating and visualizing multi-agent systems and is specially designed for testing machine learning applications (and specially focused on Case Based Reasoning ones). It includes support for representing information using the Feature Term formalism, and provides a series of relational machine learning algorithms that can deal with them. The whole project is created in C++ to maximize efficiency, and uses OpenGL as the visualization library to ensure cross-platformness.

1.6.2.2 Amaltheia

Amaltheia (http://home.gna.org/amaltheia/) is a cross-platform game programming API that supports two backends, OpenGL and DirectX. The aim of the Amaltheia project is to create an intuitive and simple to use library, providing core 3d and 2d functionality in a platform independent manner. It also provides platform independence regarding basic network functions, input handling, threads and sound. Currently the GNU/Linux and the Windows OSes are supported.

1.6.2.3 Armagetron Advanced

Armagetron Advanced (http://www.armagetronad.net/) is a multiplayer game in 3d that attempts to emulate and expand on the lightcycle sequence from the movie Tron. It's an old school arcade game slung into the 21st century. Highlights include a customizable playing arena, HUD, unique graphics, and AI bots. For the more advanced player there are new game modes and a wide variety of physics settings to tweak as well.

1.6.2.4 Audicle

Audicle (http://audicle.cs.princeton.edu/) is an audio programming environment that integrates the programmability of the development environment with elements of the runtime environment. The result is a duct-taped intersection of a concurrent smart editor, compiler, virtual machine, and debugger.

1.6.2.5 Battlestar T.U.X.

Battlestar T.U.X. (http://code.google.com/p/battlestar-tux/) is a top-down scrolling shooter project.

1.6.2.6 BJS

BJS (http://bjs.sourceforge.net/) is a funny arcade 3D multiplayer tank battle. It is fuly playable and very fun in multiplayer. Of course the single player is also possible. There is no story. You just get a tank and go shoot other players. Currently there are 5 different tanks, 6 maps, 9 powerups and 4 weapons.

1.6.2.7 Blender

Blender (http://blender.org/) is an integrated 3d suite for modelling, animation, rendering, post-production, interactive creation and playback (games).

1.6.2.8 Breve

Breve (http://www.spiderland.org/) is a free, open-source software package which makes it easy to build 3D simulations of multi-agent systems and artificial life. Using Python, or using a simple scripting language called steve, you can define the behaviors of agents in a 3D world and observe how they interact. breve includes physical simulation and collision detection so you can simulate realistic creatures, and an OpenGL display engine so you can visualize your simulated worlds.

1.6.2.9 BZFlag

BZFlag (http://BZFlag.org/) is a 3D multi-player multiplatform tank battle game that allows users to play against each other in a network environment.

BZFlag uses FTGL as of version 2.99.

1.6.2.10 Capture The Flag

Capture The Flag (http://capturetf.sourceforge.net/) is an open source, multi-platform, network game project.

1.6.2.11 Cello

Cello (http://common-lisp.net/project/cello/) is a project to create an open-source, industrial-strength, portable GUI toolkit for Common Lisp. Its features include anti-aliased fonts, accelerated 2d- and 3d-graphics, a standard set of GUI widgets, easy construction of new widgets, and much more. Cello heavily utilizes Cells (a sister project on common-lisp.net), in addition to industry-standard technologies such as OpenGL, Free↩ Type, and ImageMagick.

1.6.2.12 Chimera

Chimera (http://www.cgl.ucsf.edu/chimera/) is a highly extensible program for interactive visualization and analysis of molecular structures and related data, including density maps, supramolecular assemblies, sequence alignments, docking results, trajectories, and conformational ensembles. High-quality images and animations can be generated.

1.6.2.13 Cinepaint

Cinepaint (http://www.cinepaint.org/) is a deep paint image retouching tool that supports higher color fidelity than ordinary painting tools.

1.6.2.14 Duel

Duel (http://www.personal.rdg.ac.uk/~sir03me/play/code.html) is a small overhead perspective spaceship game.

1.6.2.15 EMAN2

EMAN2 (http://blake.bcm.tmc.edu/eman/eman2/) is a suite of scientific image processing tools aimed primarily at the transmission electron microscopy community.

1.6.2.16 EMAN2

Empty Clip (http://emptyclip.sourceforge.net/) is a top-down 2D Action RPG.

1.6.2.17 Freebox

Freebox (http://freebox.sourceforge.net/) is designed for use in a special type of computer called an 'HTPC', which is connected to a home-theatre system to watch XviD/DivX/DVD movies, play music (MP3, CD, whatever), play some emulated games, or whatever else you want to do with it.

1.6.2.18 Gem

Gem (http://gem.iem.at/) is a loadable library for puredata, which adds OpenGL graphics rendering and animation to Pd. Pd is a graphical programming language and computer music system.

1.6.2.19 GLMayan

GLMayan (http://glmayan.sourceforge.net/) is an OpenGL screensaver.

1.6.2.20 Glover

Glover (http://code.google.com/p/glover/) is a movie player that renders the content using openGL allowing all kinds of special effects using fragment shaders. The movie decoding is done using ffmpeg.

1.6.2.21 Ivf++

Ivf++ (http://ivfplusplus.sourceforge.net/) is a C++ library encapsulating OpenGL functionality. The primary goal is to make it easier to use the OpenGL library in interactive 3D applications. The second goal is extendibility, providing a set of well defined base classes for different object types to build new classes on. The third goal is portability, primarily between Linux and Windows, but the library should also be easily ported to Mac OS X.

1.6.2.22 Jahshaka

Jashaka (http://jahshaka.org/) is an advanced video editing, animation, visual effects, painting and music tool.

1.6.2.23 Karaoke FX

Karaoke FX (http://jeanchristophe.duber.free.fr/karaokefx/) is a midifile player that can display lyrics in synch whith the sound so as it can be used for karaoke. It relies on plugins for midi output devices as for lyrics display.

1.6.2.24 Libinstrudeo

Libinstrudeo (http://sourceforge.net/projects/libinstrudeo), initially written for the Screen↩ Kast program, provides the necessary logic to capture screen recordings and to process them. Includes a soap-client for the webservice at captorials.com that enables you to share your recordings.

1.6.2.25 Light Speed!

Light Speed! (http://lightspeed.sourceforge.net/) is an OpenGL-based program which illustrates the effects of special relativity on the appearance of moving objects. When an object accelerates past a few million meters per second, these effects begin to grow noticeable, becoming more and more pronounced as the speed of light is approached. These relativistic effects are viewpoint-dependent, and include shifts in length, object hue, brightness and shape.

1.6.2.26 MySQL GUI Tools

MySQL GUI Tools (http://dev.mysql.com/downloads/gui-tools/5.0.html) is a collection of tools for the MySQL database. It consists of MySQL Administrator, MySQL Query Browser and MySQL Migration Toolkit.

1.6.2.27 OctPlot

OctPlot (http://octplot.sourceforge.net/) is a graphics package for Octave, the free alternative to MATLAB. It provides high quality PostScript and on-screen graphics.

1.6.2.28 Open ActiveWrl

Open ActiveWrl (http://open-activewrl.sourceforge.net/) is a software development toolkit based on a generic software development approach that allows the implementation VRML/X3D browser componentes. These browser components can run within an conventional application or can be linked together for the implementation of parallel immersive VR setups.

1.6.2.29 OpenEaagles

OpenEaagles (http://www.openeaagles.org/) is a multi-platform simulation framework targeted to help simulation engineers and software developers build robust, scalable, virtual, constructive, stand-alone, and distributed simulation applications. It has been used extensively to build applications that demand real-time performance. This includes applications to conduct human factor studies, operator training, and the development of complete distributed virtual simulation systems. OpenEaagles has also been used to build stand-alone and distributed constructive applications oriented at system analysis.

1.6.2.30 OpenGC

OpenGC (http://www.opengc.org/) is a multi-platform, multi-simulator, open-source C++ tool for developing and implementing high quality glass cockpit displays for simulated flightdecks.

1.6.2.31 OpenSG

OpenSG (http://www.opensg.org/) is a portable scenegraph system to create realtime graphics programs, e.g. for virtual reality applications.

1.6.2.32 Panthera

Panthera (http://sourceforge.net/projects/panthera) is a C++ framework for interactive visualization, manipulation, and editing of volume data. Applications developed on top of Panthera can utilize both desktop and immersive user interface devices, such as position trackers and haptic displays.

1.6.2.33 Planet Penguin Racer

PlanetPenguin Racer (http://developer.berlios.de/projects/ppracer/) is a simple OpenGL racing game featuring Tux, the Linux mascot. The goal of the game is to slide down a snow- and ice-covered mountain as quickly as possible, avoiding the trees and rocks that will slow you down.

1.6.2.34 projectM

projectM (http://projectm.sourceforge.net/) is a music visualizer which uses OpenGL for hardware acceleration. It is compatible with Milkdrop presets.

1.6.2.35 Puzzle Bobble 3D

Puzzle Bobble 3D (http://homepage.mac.com/eric.lee/puzzle/) is a 3D video game for Linux. The game is similar to Tetris/Connect 4: connect balls of the same colour to make them disappear. Puzzle Bobble 3D is based on an already popular arcade game of the same name by Taito Corporation (see links section at the bottom of this page), but this particular variant is played in a 3D environment (hence the name).

1.6.2.36 ROOT

ROOT (http://root.cern.ch/) is an object-oriented data analysis framework.

1.6.2.37 SCIRun

SCIRun (http://software.sci.utah.edu/scirun.html) is a Problem Solving Environment (PSE), for modeling, simulation and visualization of scientific problems. It is available for free and open source.

1.6.2.38 TINE

TINE, or TINE Is Not ELITE (http://tine.sunsite.dk/en/index.html) is an open source cross-platform remake of the classic space adventure game ELITE.

1.6.2.39 Tiny Planet

Tiny Planet (http://www.duberga.net/tinyplanet/) is a real-time OpenGL viewer of detailled earth texture such as BlueMarble from Earth Observatory (NASA) or any other planet texture. Vectorial data such as points of interest, boundaries, rivers can be superimposed to the texture.

1.6.2.40 Truevision

Truevision (http://truevision.sourceforge.net/) is a 3D modeler for GNOME.

1.6.2.41 Tulip

Tulip (http://tulip.labri.fr/) is a system dedicated to the visualization of huge graphs. It is capable of managing graphs with up to 500,000 nodes and edges on relatively modest hardware (eg. 600MHz Pentium III, 256MB RAM).

1.6.2.42 Ubit

Ubit (http://www.infres.enst.fr/~elc/ubit/) Ubit is a new GUI toolkit that combines the advantages of scene graph and widget based toolkits. The Ubit3D extension makes it possible to display 2D GUIs in a 3D space.

1.6.2.43 VRS

The Virtual Rendering System (http://www.hpi.uni-potsdam.de/vrs/) is a computer graphics software library for constructing interactive 3D applications. It provides a large collection of 3D rendering components which facilitate implementing 3D graphics applications and experimenting with 3D graphics and imaging algorithms.

1.6.2.44 VTK

VTK, the Visualization Toolkit (http://www.vtk.org/), is an object oriented, high level library that allows one to easily write C++ programs, Tcl, Python and Java scripts that do 3D visualization.

1.6.2.45 XLock

XLock (http://www.tux.org/~bagleyd/xlockmore.html) is a screensaver and screen locking utility with additional OpenGL and XPM modes.

1.6.3 Projects that used to use %FTGL

1.6.3.1 GNU Backgammon

GNU Backgammon (http://www.gnubg.org/) was using FTGL until version 0.14.3+20060520-1.

1.6.3.2 OpenSceneGraph

OpenSceneGraph (http://www.openscenegraph.org/projects/osg) is an open source high performance 3D graphics toolkit, used by application developers in fields such as visual simulation, games, virtual reality, scientific visualization and modelling. Written entirely in Standard C++ and OpenGL it runs on all Windows platforms, OSX, GNU/Linux, IRIX, Solaris, HP-Ux, AIX and FreeBSD operating systems.

1.6.3.3 Teddy

Teddy (http://teddy.sourceforge.net/) was a 3D graphics library. The main purpose was to be a simple scene graph manager.

1.6.3.4 VigiPac

VigiPac (http://vigipac.sourceforge.net/) was a three-dimensional Pacman clone with multiplayer support, written in the C++ language.

Chapter 2

Namespace Documentation

2.1 FTGL Namespace Reference

Enumerations

- enum RenderMode { RENDER_FRONT = 0x0001 , RENDER_BACK = 0x0002 , RENDER_SIDE = 0x0004 , RENDER_ALL = 0xffff }
- enum TextAlignment { ALIGN_LEFT = 0 , ALIGN_CENTER = 1 , ALIGN_RIGHT = 2 , ALIGN_JUSTIFY = 3 }
- enum ConfigString { CONFIG_VERSION = 1 }

Functions

- char const * GetString (ConfigString config)

  Return a string describing the current FTGL instance.

2.1.1 Enumeration Type Documentation

2.1.1.1 ConfigString

`enum FTGL::ConfigString`

Enumerator

CONFIG_VERSION	

Definition at line 69 of file ftgl.h.

2.1.1.2 RenderMode

`enum FTGL::RenderMode`

Enumerator

RENDER_FRONT	
RENDER_BACK	
RENDER_SIDE	
RENDER_ALL	

Definition at line 53 of file ftgl.h.

2.1.1.3 TextAlignment

```
enum FTGL::TextAlignment
```

Enumerator

ALIGN_LEFT	
ALIGN_CENTER	
ALIGN_RIGHT	
ALIGN_JUSTIFY	

Definition at line 61 of file ftgl.h.

2.1.2 Function Documentation

2.1.2.1 GetString()

```
char const * FTGL::GetString (
            ConfigString config )
```

Return a string describing the current FTGL instance.

Parameters

config	Name of the string to retrieve. Can be one of: • CONFIG_VERSION: return the FTGL release number.

Returns

A pointer to a constant string containing the requested information, or 0 in case of invalid argument.

Chapter 3

Data Structure Documentation

3.1 FTBBox Class Reference

FTBBox is a convenience class for handling bounding boxes.

`#include <FTBBox.h>`

Public Member Functions

- FTBBox ()
  Default constructor.
- FTBBox (float lx, float ly, float lz, float ux, float uy, float uz)
  Constructor.
- FTBBox (FTPoint l, FTPoint u)
  Constructor.
- FTBBox (FT_GlyphSlot glyph)
  Constructor.
- ~FTBBox ()
  Destructor.
- void Invalidate ()
  Mark the bounds invalid by setting all lower dimensions greater than the upper dimensions.
- bool IsValid ()
  Determines if this bounding box is valid.
- FTBBox & operator+= (const FTPoint vector)
  Move the Bounding Box by a vector.
- FTBBox & operator|= (const FTBBox &bbox)
  Combine two bounding boxes.
- void SetDepth (float depth)
- FTPoint const Upper () const
- FTPoint const Lower () const

3.1.1 Detailed Description

FTBBox is a convenience class for handling bounding boxes.

Definition at line 42 of file FTBBox.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 FTBBox() [1/4]

```
FTBBox::FTBBox ( )  [inline]
```

Default constructor.

Bounding box is set to zero.

Definition at line 48 of file FTBBox.h.

3.1.2.2 FTBBox() [2/4]

```
FTBBox::FTBBox (
            float lx,
            float ly,
            float lz,
            float ux,
            float uy,
            float uz )  [inline]
```

Constructor.

Definition at line 56 of file FTBBox.h.

3.1.2.3 FTBBox() [3/4]

```
FTBBox::FTBBox (
            FTPoint l,
            FTPoint u )  [inline]
```

Constructor.

Definition at line 64 of file FTBBox.h.

3.1.2.4 FTBBox() [4/4]

```
FTBBox::FTBBox (
            FT_GlyphSlot glyph )  [inline]
```

Constructor.

Extracts a bounding box from a freetype glyph. Uses the control box for the glyph. `FT_Glyph_Get_CBox()`

Parameters

glyph	A freetype glyph

Definition at line 75 of file FTBBox.h.

3.1.2.5 ~FTBBox()

```
FTBBox::~FTBBox ( )  [inline]
```

Destructor.

Definition at line 93 of file FTBBox.h.

3.1.3 Member Function Documentation

3.1.3.1 Invalidate()

```
void FTBBox::Invalidate ( )  [inline]
```

Mark the bounds invalid by setting all lower dimensions greater than the upper dimensions.

Definition at line 100 of file FTBBox.h.

3.1.3.2 IsValid()

```
bool FTBBox::IsValid ( )  [inline]
```

Determines if this bounding box is valid.

Returns

True if all lower values are <= the corresponding upper values.

Definition at line 112 of file FTBBox.h.

3.1.3.3 Lower()

```
FTPoint const FTBBox::Lower ( ) const  [inline]
```

Definition at line 165 of file FTBBox.h.

Referenced by FTFont::BBox(), and FTFont::BBox().

3.1.3.4 operator+=()

```
FTBBox & FTBBox::operator+= (
            const FTPoint vector )  [inline]
```

Move the Bounding Box by a vector.

Parameters

vector	The vector to move the bbox in 3D space.

Definition at line 124 of file FTBBox.h.

3.1.3.5 operator"|=()

```
FTBBox & FTBBox::operator|= (
            const FTBBox & bbox )  [inline]
```

Combine two bounding boxes.

The result is the smallest bounding box containing the two original boxes.

Parameters

bbox	The bounding box to merge with the second one.

Definition at line 138 of file FTBBox.h.

References FTPoint::X(), FTPoint::Y(), and FTPoint::Z().

3.1.3.6 SetDepth()

```
void FTBBox::SetDepth (
            float depth )  [inline]
```

Definition at line 150 of file FTBBox.h.

3.1.3.7 Upper()

```
FTPoint const FTBBox::Upper ( ) const  [inline]
```

Definition at line 159 of file FTBBox.h.

Referenced by FTFont::BBox(), and FTFont::BBox().

The documentation for this class was generated from the following file:

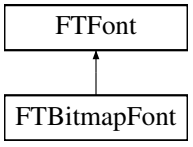
- FTBBox.h

3.2 FTBitmapFont Class Reference

FTBitmapFont is a specialisation of the FTFont class for handling Bitmap fonts.

```
#include <FTGLBitmapFont.h>
```

Inheritance diagram for FTBitmapFont:



Public Member Functions

- FTBitmapFont (const char *fontFilePath)

  Open and read a font file.
- FTBitmapFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.
- ~FTBitmapFont ()

  Destructor.

Public Member Functions inherited from FTFont

- virtual ~FTFont ()
- virtual bool Attach (const char *fontFilePath)

  Attach auxilliary file to font e.g font metrics.
- virtual bool Attach (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Attach auxilliary data to font e.g font metrics, from memory.
- virtual void GlyphLoadFlags (FT_Int flags)

  Set the glyph loading flags.
- virtual bool CharMap (FT_Encoding encoding)

  Set the character map for the face.
- virtual unsigned int CharMapCount () const

  Get the number of character maps in this face.
- virtual FT_Encoding * CharMapList ()

  Get a list of character maps in this face.
- virtual bool FaceSize (const unsigned int size, const unsigned int res=72)

  Set the char size for the current face.
- virtual unsigned int FaceSize () const

  Get the current face size in points (1/72 inch).
- virtual void Depth (float depth)

  Set the extrusion distance for the font.
- virtual void Outset (float outset)

  Set the outset distance for the font.
- virtual void Outset (float front, float back)

  Set the front and back outset distances for the font.
- virtual void UseDisplayList (bool useList)

  Enable or disable the use of Display Lists inside FTGL.
- virtual float Ascender () const

  Get the global ascender height for the face.
- virtual float Descender () const

  Gets the global descender height for the face.
- virtual float LineHeight () const

  Gets the line spacing for the font.
- virtual FTBBox BBox (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const char *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).
- virtual FTBBox BBox (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const wchar_t *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

Get the bounding box for a string (deprecated).

- virtual float Advance (const char *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual float Advance (const wchar_t *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual FTPoint Render (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FTPoint Render (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FT_Error Error () const

  Queries the Font for errors.

Protected Member Functions

- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)

  Construct a glyph of the correct type.

Protected Member Functions inherited from FTFont

- FTFont (char const *fontFilePath)

  Open and read a font file.
- FTFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.
- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)=0

  Construct a glyph of the correct type.

3.2.1 Detailed Description

FTBitmapFont is a specialisation of the FTFont class for handling Bitmap fonts.

See also

FTFont

Definition at line 45 of file FTGLBitmapFont.h.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 FTBitmapFont() [1/2]

```
FTBitmapFont::FTBitmapFont (
            const char * fontFilePath )
```

Open and read a font file.

Sets Error flag.

Parameters

fontFilePath	font file path.

3.2.2.2 FTBitmapFont() [2/2]

```
FTBitmapFont::FTBitmapFont (
            const unsigned char * pBufferBytes,
            size_t bufferSizeInBytes )
```

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

pBufferBytes	the in-memory buffer
bufferSizeInBytes	the length of the buffer in bytes

3.2.2.3 ~FTBitmapFont()

```
FTBitmapFont::~FTBitmapFont ( )
```

Destructor.

3.2.3 Member Function Documentation

3.2.3.1 MakeGlyph()

```
virtual FTGlyph * FTBitmapFont::MakeGlyph (
            FT_GlyphSlot slot )  [protected], [virtual]
```

Construct a glyph of the correct type.

Clients must override the function and return their specialised FTGlyph.

Parameters

slot	A FreeType glyph slot.

Returns

An FT****Glyph or `null` on failure.

Implements FTFont.

The documentation for this class was generated from the following file:

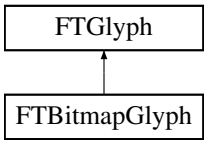
- FTGLBitmapFont.h

3.3 FTBitmapGlyph Class Reference

FTBitmapGlyph is a specialisation of FTGlyph for creating bitmaps.

`#include <FTBitmapGlyph.h>`

Inheritance diagram for FTBitmapGlyph:



Public Member Functions

- FTBitmapGlyph (FT_GlyphSlot glyph)

  Constructor.
- virtual ~FTBitmapGlyph ()

  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)

  Render this glyph at the current pen position.

Public Member Functions inherited from FTGlyph

- virtual ~FTGlyph ()

  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)=0

  Renders this glyph at the current pen position.
- virtual float Advance () const

  Return the advance width for this glyph.
- virtual const FTBBox & BBox () const

  Return the bounding box for this glyph.
- virtual FT_Error Error () const

  Queries for errors.

Additional Inherited Members

Protected Member Functions inherited from FTGlyph

- FTGlyph (FT_GlyphSlot glyph)

  Create a glyph.

3.3.1 Detailed Description

FTBitmapGlyph is a specialisation of FTGlyph for creating bitmaps.

Definition at line 42 of file FTBitmapGlyph.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 FTBitmapGlyph()

```
FTBitmapGlyph::FTBitmapGlyph (
            FT_GlyphSlot glyph )
```

Constructor.

Parameters

glyph	The Freetype glyph to be processed

3.3.2.2 ~FTBitmapGlyph()

```
virtual FTBitmapGlyph::~FTBitmapGlyph ( )  [virtual]
```

Destructor.

3.3.3 Member Function Documentation

3.3.3.1 Render()

```
virtual const FTPoint & FTBitmapGlyph::Render (
            const FTPoint & pen,
            int renderMode )  [virtual]
```

Render this glyph at the current pen position.

Parameters

pen	The current pen position.
renderMode	Render mode to display

Returns

The advance distance for this glyph.

Implements FTGlyph.

The documentation for this class was generated from the following file:

- FTBitmapGlyph.h

3.4 FTBuffer Class Reference

FTBuffer is a helper class for pixel buffers.

`#include <FTBuffer.h>`

Public Member Functions

- FTBuffer ()
  Default constructor.
- ~FTBuffer ()
  Destructor.
- FTPoint Pos () const
  Get the pen's position in the buffer.
- void Pos (FTPoint arg)
  Set the pen's position in the buffer.
- void Size (int w, int h)
  Set the buffer's size.
- int Width () const
  Get the buffer's width.
- int Height () const
  Get the buffer's height.
- unsigned char * Pixels () const
  Get the buffer's direct pixel buffer.

3.4.1 Detailed Description

FTBuffer is a helper class for pixel buffers.

It provides the interface between FTBufferFont and FTBufferGlyph to optimise rendering operations.

See also

FTBufferGlyph
FTBufferFont

Definition at line 45 of file FTBuffer.h.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 FTBuffer()

`FTBuffer::FTBuffer ( )`

Default constructor.

3.4.2.2 ~FTBuffer()

```
FTBuffer::~FTBuffer ( )
```

Destructor.

3.4.3 Member Function Documentation

3.4.3.1 Height()

```
int FTBuffer::Height ( ) const  [inline]
```

Get the buffer's height.

Returns

The buffer's height, in pixels.

Definition at line 98 of file FTBuffer.h.

3.4.3.2 Pixels()

```
unsigned char * FTBuffer::Pixels ( ) const  [inline]
```

Get the buffer's direct pixel buffer.

Returns

A read-write pointer to the buffer's pixels.

Definition at line 105 of file FTBuffer.h.

3.4.3.3 Pos() [1/2]

```
FTPoint FTBuffer::Pos ( ) const  [inline]
```

Get the pen's position in the buffer.

Returns

The pen's position as an FTPoint object.

Definition at line 63 of file FTBuffer.h.

3.4.3.4 Pos() [2/2]

```
void FTBuffer::Pos (
            FTPoint arg )  [inline]
```

Set the pen's position in the buffer.

Parameters

arg	An FTPoint object with the desired pen's position.

Definition at line 73 of file FTBuffer.h.

3.4.3.5 Size()

```
void FTBuffer::Size (
            int w,
            int h )
```

Set the buffer's size.

Parameters

w	The buffer's desired width, in pixels.
h	The buffer's desired height, in pixels.

3.4.3.6 Width()

```
int FTBuffer::Width ( ) const  [inline]
```

Get the buffer's width.

Returns

The buffer's width, in pixels.

Definition at line 91 of file FTBuffer.h.

The documentation for this class was generated from the following file:

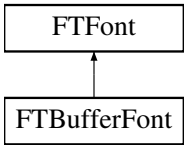
- FTBuffer.h

3.5 FTBufferFont Class Reference

FTBufferFont is a specialisation of the FTFont class for handling memory buffer fonts.

```
#include <FTBufferFont.h>
```

Inheritance diagram for FTBufferFont:



Public Member Functions

- FTBufferFont (const char *fontFilePath)

  Open and read a font file.
- FTBufferFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.
- ~FTBufferFont ()

  Destructor.

Public Member Functions inherited from FTFont

- virtual ~FTFont ()
- virtual bool Attach (const char *fontFilePath)

  Attach auxilliary file to font e.g font metrics.
- virtual bool Attach (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Attach auxilliary data to font e.g font metrics, from memory.
- virtual void GlyphLoadFlags (FT_Int flags)

  Set the glyph loading flags.
- virtual bool CharMap (FT_Encoding encoding)

  Set the character map for the face.
- virtual unsigned int CharMapCount () const

  Get the number of character maps in this face.
- virtual FT_Encoding * CharMapList ()

  Get a list of character maps in this face.
- virtual bool FaceSize (const unsigned int size, const unsigned int res=72)

  Set the char size for the current face.
- virtual unsigned int FaceSize () const

  Get the current face size in points (1/72 inch).
- virtual void Depth (float depth)

  Set the extrusion distance for the font.
- virtual void Outset (float outset)

  Set the outset distance for the font.
- virtual void Outset (float front, float back)

  Set the front and back outset distances for the font.
- virtual void UseDisplayList (bool useList)

  Enable or disable the use of Display Lists inside FTGL.
- virtual float Ascender () const

  Get the global ascender height for the face.
- virtual float Descender () const

  Gets the global descender height for the face.
- virtual float LineHeight () const

  Gets the line spacing for the font.
- virtual FTBBox BBox (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const char *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).
- virtual FTBBox BBox (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const wchar_t *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

Get the bounding box for a string (deprecated).

- virtual float Advance (const char *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual float Advance (const wchar_t *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual FTPoint Render (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FTPoint Render (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FT_Error Error () const

  Queries the Font for errors.

Protected Member Functions

- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)

  Construct a glyph of the correct type.

Protected Member Functions inherited from FTFont

- FTFont (char const *fontFilePath)

  Open and read a font file.
- FTFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.
- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)=0

  Construct a glyph of the correct type.

3.5.1 Detailed Description

FTBufferFont is a specialisation of the FTFont class for handling memory buffer fonts.

See also

FTFont

Definition at line 43 of file FTBufferFont.h.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 FTBufferFont() [1/2]

```
FTBufferFont::FTBufferFont (
            const char * fontFilePath )
```

Open and read a font file.

Sets Error flag.

Parameters

fontFilePath	font file path.

3.5.2.2 FTBufferFont() [2/2]

```
FTBufferFont::FTBufferFont (
            const unsigned char * pBufferBytes,
            size_t bufferSizeInBytes )
```

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

pBufferBytes	the in-memory buffer
bufferSizeInBytes	the length of the buffer in bytes

3.5.2.3 ~FTBufferFont()

```
FTBufferFont::~FTBufferFont ( )
```

Destructor.

3.5.3 Member Function Documentation

3.5.3.1 MakeGlyph()

```
virtual FTGlyph * FTBufferFont::MakeGlyph (
            FT_GlyphSlot slot )  [protected], [virtual]
```

Construct a glyph of the correct type.

Clients must override the function and return their specialised FTGlyph.

Parameters

slot	A FreeType glyph slot.

Returns

An FT****Glyph or `null` on failure.

Implements FTFont.

The documentation for this class was generated from the following file:

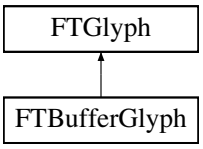
- FTBufferFont.h

3.6 FTBufferGlyph Class Reference

FTBufferGlyph is a specialisation of FTGlyph for memory buffer rendering.

`#include <FTBufferGlyph.h>`

Inheritance diagram for FTBufferGlyph:



Public Member Functions

- FTBufferGlyph (FT_GlyphSlot glyph, FTBuffer *buffer)

  Constructor.
- virtual ~FTBufferGlyph ()

  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)

  Render this glyph at the current pen position.

Public Member Functions inherited from FTGlyph

- virtual ~FTGlyph ()

  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)=0

  Renders this glyph at the current pen position.
- virtual float Advance () const

  Return the advance width for this glyph.
- virtual const FTBBox & BBox () const

  Return the bounding box for this glyph.
- virtual FT_Error Error () const

  Queries for errors.

Additional Inherited Members

Protected Member Functions inherited from FTGlyph

- FTGlyph (FT_GlyphSlot glyph)

  Create a glyph.

3.6.1 Detailed Description

FTBufferGlyph is a specialisation of FTGlyph for memory buffer rendering.

Definition at line 40 of file FTBufferGlyph.h.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 FTBufferGlyph()

```
FTBufferGlyph::FTBufferGlyph (
            FT_GlyphSlot glyph,
            FTBuffer * buffer )
```

Constructor.

Parameters

glyph	The Freetype glyph to be processed
buffer	An FTBuffer object in which to render the glyph.

3.6.2.2 ~FTBufferGlyph()

```
virtual FTBufferGlyph::~FTBufferGlyph ( )  [virtual]
```

Destructor.

3.6.3 Member Function Documentation

3.6.3.1 Render()

```
virtual const FTPoint & FTBufferGlyph::Render (
            const FTPoint & pen,
            int renderMode )  [virtual]
```

Render this glyph at the current pen position.

Parameters

pen	The current pen position.
renderMode	Render mode to display

Returns

The advance distance for this glyph.

Implements FTGlyph.

The documentation for this class was generated from the following file:

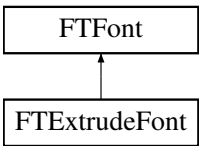
- FTBufferGlyph.h

3.7 FTExtrudeFont Class Reference

FTExtrudeFont is a specialisation of the FTFont class for handling extruded Polygon fonts.

`#include <FTGLExtrdFont.h>`

Inheritance diagram for FTExtrudeFont:



Public Member Functions

- FTExtrudeFont (const char *fontFilePath)

  Open and read a font file.
- FTExtrudeFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.
- ~FTExtrudeFont ()

  Destructor.

Public Member Functions inherited from FTFont

- virtual ~FTFont ()
- virtual bool Attach (const char *fontFilePath)

  Attach auxilliary file to font e.g font metrics.
- virtual bool Attach (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Attach auxilliary data to font e.g font metrics, from memory.
- virtual void GlyphLoadFlags (FT_Int flags)

  Set the glyph loading flags.
- virtual bool CharMap (FT_Encoding encoding)

  Set the character map for the face.
- virtual unsigned int CharMapCount () const

  Get the number of character maps in this face.
- virtual FT_Encoding * CharMapList ()

  Get a list of character maps in this face.
- virtual bool FaceSize (const unsigned int size, const unsigned int res=72)

  Set the char size for the current face.
- virtual unsigned int FaceSize () const

  Get the current face size in points (1/72 inch).
- virtual void Depth (float depth)

  Set the extrusion distance for the font.

- virtual void Outset (float outset)

  Set the outset distance for the font.
- virtual void Outset (float front, float back)

  Set the front and back outset distances for the font.
- virtual void UseDisplayList (bool useList)

  Enable or disable the use of Display Lists inside FTGL.
- virtual float Ascender () const

  Get the global ascender height for the face.
- virtual float Descender () const

  Gets the global descender height for the face.
- virtual float LineHeight () const

  Gets the line spacing for the font.
- virtual FTBBox BBox (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const char *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).
- virtual FTBBox BBox (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const wchar_t *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).
- virtual float Advance (const char *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual float Advance (const wchar_t *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual FTPoint Render (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FTPoint Render (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FT_Error Error () const

  Queries the Font for errors.

Protected Member Functions

- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)

  Construct a glyph of the correct type.

Protected Member Functions inherited from FTFont

- FTFont (char const *fontFilePath)

  Open and read a font file.
- FTFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.
- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)=0

  Construct a glyph of the correct type.

3.7.1 Detailed Description

FTExtrudeFont is a specialisation of the FTFont class for handling extruded Polygon fonts.

See also

FTFont

FTPolygonFont

Definition at line 46 of file FTGLExtrdFont.h.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 FTExtrudeFont() [1/2]

```
FTExtrudeFont::FTExtrudeFont (
            const char * fontFilePath )
```

Open and read a font file.

Sets Error flag.

Parameters

fontFilePath	font file path.

3.7.2.2 FTExtrudeFont() [2/2]

```
FTExtrudeFont::FTExtrudeFont (
            const unsigned char * pBufferBytes,
            size_t bufferSizeInBytes )
```

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

pBufferBytes	the in-memory buffer
bufferSizeInBytes	the length of the buffer in bytes

3.7.2.3 ~FTExtrudeFont()

```
FTExtrudeFont::~FTExtrudeFont ( )
```

Destructor.

3.7.3 Member Function Documentation

3.7.3.1 MakeGlyph()

```
virtual FTGlyph * FTExtrudeFont::MakeGlyph (
            FT_GlyphSlot slot )  [protected], [virtual]
```

Construct a glyph of the correct type.

Clients must override the function and return their specialised FTGlyph.

Parameters

slot	A FreeType glyph slot.

Returns

An FT****Glyph or `null` on failure.

Implements FTFont.

The documentation for this class was generated from the following file:

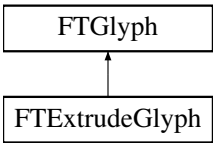
- FTGLExtrdFont.h

3.8 FTExtrudeGlyph Class Reference

FTExtrudeGlyph is a specialisation of FTGlyph for creating tessellated extruded polygon glyphs.

```
#include <FTExtrdGlyph.h>
```

Inheritance diagram for FTExtrudeGlyph:



Public Member Functions

- FTExtrudeGlyph (FT_GlyphSlot glyph, float depth, float frontOutset, float backOutset, bool useDisplayList)
  Constructor.
- virtual ~FTExtrudeGlyph ()
  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)
  Render this glyph at the current pen position.

Public Member Functions inherited from FTGlyph

- virtual ~FTGlyph ()

  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)=0

  Renders this glyph at the current pen position.
- virtual float Advance () const

  Return the advance width for this glyph.
- virtual const FTBBox & BBox () const

  Return the bounding box for this glyph.
- virtual FT_Error Error () const

  Queries for errors.

Additional Inherited Members

Protected Member Functions inherited from FTGlyph

- FTGlyph (FT_GlyphSlot glyph)

  Create a glyph.

3.8.1 Detailed Description

FTExtrudeGlyph is a specialisation of FTGlyph for creating tessellated extruded polygon glyphs.

Definition at line 43 of file FTExtrdGlyph.h.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 FTExtrudeGlyph()

```
FTExtrudeGlyph::FTExtrudeGlyph (
            FT_GlyphSlot glyph,
            float depth,
            float frontOutset,
            float backOutset,
            bool useDisplayList )
```

Constructor.

Sets the Error to Invalid_Outline if the glyph isn't an outline.

Parameters

glyph	The Freetype glyph to be processed
depth	The distance along the z axis to extrude the glyph
frontOutset	outset contour size
backOutset	outset contour size
useDisplayList	Enable or disable the use of Display Lists for this glyph `true` turns ON display lists. `false` turns OFF display lists.

3.8.2.2 ~FTExtrudeGlyph()

```
virtual FTExtrudeGlyph::~FTExtrudeGlyph ( )  [virtual]
```

Destructor.

3.8.3 Member Function Documentation

3.8.3.1 Render()

```
virtual const FTPoint & FTExtrudeGlyph::Render (
            const FTPoint & pen,
            int renderMode )  [virtual]
```

Render this glyph at the current pen position.

Parameters

pen	The current pen position.
renderMode	Render mode to display

Returns

The advance distance for this glyph.

Implements FTGlyph.

The documentation for this class was generated from the following file:

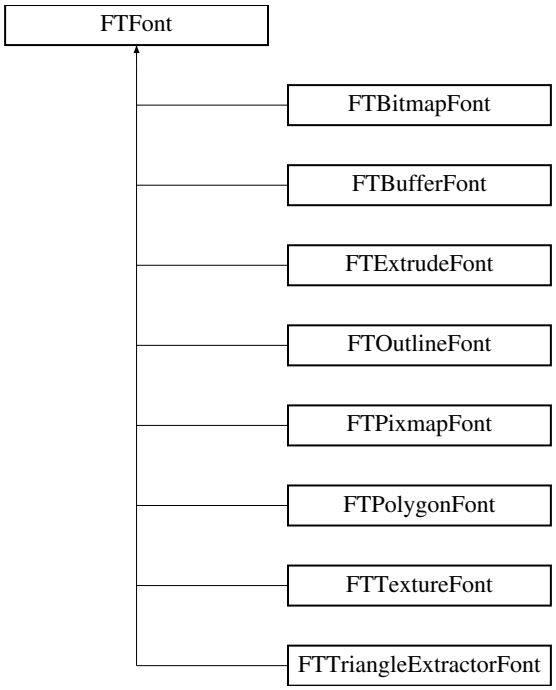
- FTExtrdGlyph.h

3.9 FTFont Class Reference

FTFont is the public interface for the FTGL library.

```
#include <FTFont.h>
```

Inheritance diagram for FTFont:



Public Member Functions

- virtual ~FTFont ()
- virtual bool Attach (const char *fontFilePath)

  Attach auxilliary file to font e.g font metrics.
- virtual bool Attach (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Attach auxilliary data to font e.g font metrics, from memory.
- virtual void GlyphLoadFlags (FT_Int flags)

  Set the glyph loading flags.
- virtual bool CharMap (FT_Encoding encoding)

  Set the character map for the face.
- virtual unsigned int CharMapCount () const

  Get the number of character maps in this face.
- virtual FT_Encoding * CharMapList ()

  Get a list of character maps in this face.
- virtual bool FaceSize (const unsigned int size, const unsigned int res=72)

  Set the char size for the current face.
- virtual unsigned int FaceSize () const

  Get the current face size in points (1/72 inch).
- virtual void Depth (float depth)

  Set the extrusion distance for the font.
- virtual void Outset (float outset)

  Set the outset distance for the font.
- virtual void Outset (float front, float back)

  Set the front and back outset distances for the font.
- virtual void UseDisplayList (bool useList)

  Enable or disable the use of Display Lists inside FTGL.
- virtual float Ascender () const

  Get the global ascender height for the face.
- virtual float Descender () const

Gets the global descender height for the face.

- virtual float LineHeight () const

  Gets the line spacing for the font.

- virtual FTBBox BBox (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.

- void BBox (const char *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).

- virtual FTBBox BBox (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.

- void BBox (const wchar_t *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).

- virtual float Advance (const char *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.

- virtual float Advance (const wchar_t *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.

- virtual FTPoint Render (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.

- virtual FTPoint Render (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.

- virtual FT_Error Error () const

  Queries the Font for errors.

Protected Member Functions

- FTFont (char const *fontFilePath)

  Open and read a font file.

- FTFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.

- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)=0

  Construct a glyph of the correct type.

Friends

- class FTBitmapFont
- class FTBufferFont
- class FTExtrudeFont
- class FTOutlineFont
- class FTPixmapFont
- class FTPolygonFont
- class FTTextureFont
- class FTTriangleExtractorFont
- class FTFontImpl

3.9.1 Detailed Description

FTFont is the public interface for the FTGL library.

Specific font classes are derived from this class. It uses the helper classes FTFace and FTSize to access the Freetype library. This class is abstract and deriving classes must implement the protected `MakeGlyph` function to create glyphs of the appropriate type.

It is good practice after using these functions to test the error code returned. `FT_Error Error()`. Check the freetype file fterrdef.h for error definitions.

See also

FTFace

FTSize

Definition at line 56 of file FTFont.h.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 FTFont() [1/2]

```
FTFont::FTFont (
            char const * fontFilePath )  [protected]
```

Open and read a font file.

Sets Error flag.

Parameters

fontFilePath	font file path.

3.9.2.2 FTFont() [2/2]

```
FTFont::FTFont (
            const unsigned char * pBufferBytes,
            size_t bufferSizeInBytes )  [protected]
```

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

pBufferBytes	the in-memory buffer
bufferSizeInBytes	the length of the buffer in bytes

3.9.2.3 ~FTFont()

```
virtual FTFont::~FTFont ( )  [virtual]
```

3.9.3 Member Function Documentation

3.9.3.1 Advance() [1/2]

```
virtual float FTFont::Advance (
            const char * string,
            const int len = -1,
            FTPoint spacing = FTPoint() )  [virtual]
```

Get the advance for a string.

Parameters

string	'C' style string to be checked.
len	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
spacing	A displacement vector to add after each character has been checked (optional).

Returns

The string's advance width.

3.9.3.2 Advance() [2/2]

```
virtual float FTFont::Advance (
            const wchar_t * string,
            const int len = -1,
            FTPoint spacing = FTPoint() )  [virtual]
```

Get the advance for a string.

Parameters

string	A wchar_t string
len	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
spacing	A displacement vector to add after each character has been checked (optional).

Returns

The string's advance width.

3.9.3.3 Ascender()

```
virtual float FTFont::Ascender ( ) const  [virtual]
```

Get the global ascender height for the face.

Returns

Ascender height

3.9.3.4 Attach() [1/2]

```
virtual bool FTFont::Attach (
            const char * fontFilePath )  [virtual]
```

Attach auxilliary file to font e.g font metrics.

Note: not all font formats implement this function.

Parameters

fontFilePath	auxilliary font file path.

Returns

`true` if file has been attached successfully.

3.9.3.5 Attach() [2/2]

```
virtual bool FTFont::Attach (
            const unsigned char * pBufferBytes,
            size_t bufferSizeInBytes )  [virtual]
```

Attach auxilliary data to font e.g font metrics, from memory.

Note: not all font formats implement this function.

Parameters

pBufferBytes	the in-memory buffer.
bufferSizeInBytes	the length of the buffer in bytes.

Returns

`true` if file has been attached successfully.

3.9.3.6 BBox() [1/4]

```
virtual FTBBox FTFont::BBox (
            const char * string,
            const int len = -1,
```

```
            FTPoint position = FTPoint(),
            FTPoint spacing = FTPoint() )  [virtual]
```

Get the bounding box for a string.

Parameters

string	A char buffer.
len	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
position	The pen position of the first character (optional).
spacing	A displacement vector to add after each character has been checked (optional).

Returns

The corresponding bounding box.

3.9.3.7 BBox() [2/4]

```
void FTFont::BBox (
            const char * string,
            float & llx,
            float & lly,
            float & llz,
            float & urx,
            float & ury,
            float & urz )  [inline]
```

Get the bounding box for a string (deprecated).

Parameters

string	A char buffer.
llx	Lower left near x coordinate.
lly	Lower left near y coordinate.
llz	Lower left near z coordinate.
urx	Upper right far x coordinate.
ury	Upper right far y coordinate.
urz	Upper right far z coordinate.

Definition at line 252 of file FTFont.h.

References FTBBox::Lower(), FTBBox::Upper(), FTPoint::Xf(), FTPoint::Yf(), and FTPoint::Zf().

3.9.3.8 BBox() [3/4]

```
virtual FTBBox FTFont::BBox (
            const wchar_t * string,
            const int len = -1,
```

```
            FTPoint position = FTPoint(),
            FTPoint spacing = FTPoint() )  [virtual]
```

Get the bounding box for a string.

Parameters

string	A wchar_t buffer.
len	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
position	The pen position of the first character (optional).
spacing	A displacement vector to add after each character has been checked (optional).

Returns

The corresponding bounding box.

3.9.3.9 BBox() [4/4]

```
void FTFont::BBox (
            const wchar_t * string,
            float & llx,
            float & lly,
            float & llz,
            float & urx,
            float & ury,
            float & urz )  [inline]
```

Get the bounding box for a string (deprecated).

Parameters

string	A wchar_t buffer.
llx	Lower left near x coordinate.
lly	Lower left near y coordinate.
llz	Lower left near z coordinate.
urx	Upper right far x coordinate.
ury	Upper right far y coordinate.
urz	Upper right far z coordinate.

Definition at line 287 of file FTFont.h.

References FTBBox::Lower(), FTBBox::Upper(), FTPoint::Xf(), FTPoint::Yf(), and FTPoint::Zf().

3.9.3.10 CharMap()

```
virtual bool FTFont::CharMap (
            FT_Encoding encoding )  [virtual]
```

Set the character map for the face.

Parameters

encoding	Freetype enumerate for char map code.

Returns

`true` if charmap was valid and set correctly.

3.9.3.11 CharMapCount()

```
virtual unsigned int FTFont::CharMapCount ( ) const  [virtual]
```

Get the number of character maps in this face.

Returns

character map count.

3.9.3.12 CharMapList()

```
virtual FT_Encoding * FTFont::CharMapList ( )  [virtual]
```

Get a list of character maps in this face.

Returns

pointer to the first encoding.

3.9.3.13 Depth()

```
virtual void FTFont::Depth (
            float depth )  [virtual]
```

Set the extrusion distance for the font.

Only implemented by FTExtrudeFont

Parameters

depth	The extrusion distance.

3.9.3.14 Descender()

```
virtual float FTFont::Descender ( ) const  [virtual]
```

Gets the global descender height for the face.

Returns

Descender height

3.9.3.15 Error()

```
virtual FT_Error FTFont::Error ( ) const  [virtual]
```

Queries the Font for errors.

Returns

The current error code.

3.9.3.16 FaceSize() [1/2]

```
virtual unsigned int FTFont::FaceSize ( ) const  [virtual]
```

Get the current face size in points (1/72 inch).

Returns

face size

3.9.3.17 FaceSize() [2/2]

```
virtual bool FTFont::FaceSize (
            const unsigned int size,
            const unsigned int res = 72 )  [virtual]
```

Set the char size for the current face.

Parameters

size	the face size in points (1/72 inch)
res	the resolution of the target device.

Returns

`true` if size was set correctly

3.9.3.18 GlyphLoadFlags()

```
virtual void FTFont::GlyphLoadFlags (
            FT_Int flags )  [virtual]
```

Set the glyph loading flags.

By default, fonts use the most sensible flags when loading a font's glyph using FT_Load_Glyph(). This function allows to override the default flags.

Parameters

flags	The glyph loading flags.

3.9.3.19 LineHeight()

```
virtual float FTFont::LineHeight ( ) const  [virtual]
```

Gets the line spacing for the font.

Returns

Line height

3.9.3.20 MakeGlyph()

```
virtual FTGlyph * FTFont::MakeGlyph (
            FT_GlyphSlot slot )  [protected], [pure virtual]
```

Construct a glyph of the correct type.

Clients must override the function and return their specialised FTGlyph.

Parameters

slot	A FreeType glyph slot.

Returns

An FT****Glyph or `null` on failure.

Implemented in FTBufferFont, FTBitmapFont, FTExtrudeFont, FTOutlineFont, FTPixmapFont, FTPolygonFont, FTTextureFont, and FTTriangleExtractorFont.

3.9.3.21 Outset() [1/2]

```
virtual void FTFont::Outset (
            float front,
            float back )  [virtual]
```

Set the front and back outset distances for the font.

Only implemented by FTExtrudeFont

Parameters

front	The front outset distance.
back	The back outset distance.

3.9.3.22 Outset() [2/2]

```
virtual void FTFont::Outset (
            float outset )  [virtual]
```

Set the outset distance for the font.

Only implemented by FTOutlineFont, FTPolygonFont and FTExtrudeFont

Parameters

outset	The outset distance.

3.9.3.23 Render() [1/2]

```
virtual FTPoint FTFont::Render (
            const char * string,
            const int len = -1,
            FTPoint position = FTPoint(),
            FTPoint spacing = FTPoint(),
            int renderMode = FTGL::RENDER_ALL )  [virtual]
```

Render a string of characters.

Parameters

string	'C' style string to be output.
len	The length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).
position	The pen position of the first character (optional).
spacing	A displacement vector to add after each character has been displayed (optional).
renderMode	Render mode to use for display (optional). Gives the application the ability to control whether to render the font's front and back faces via FTGL::RENDER_FRONT and FTGL::RENDER_BACK respectively, or font sides via FTGL::RENDER_SIDE for extruded glyph fonts.

Returns

The new pen position after the last character was output.

3.9.3.24 Render() [2/2]

```
virtual FTPoint FTFont::Render (
            const wchar_t * string,
            const int len = -1,
            FTPoint position = FTPoint(),
            FTPoint spacing = FTPoint(),
            int renderMode = FTGL::RENDER_ALL )  [virtual]
```

Render a string of characters.

Parameters

string	wchar_t string to be output.
len	The length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).
position	The pen position of the first character (optional).
spacing	A displacement vector to add after each character has been displayed (optional).
renderMode	Render mode to use for display (optional). Gives the application the ability to control whether to render the font's front and back faces via FTGL::RENDER_FRONT and FTGL::RENDER_BACK respectively, or font sides via FTGL::RENDER_SIDE for extruded glyph fonts.

Returns

The new pen position after the last character was output.

3.9.3.25 UseDisplayList()

```
virtual void FTFont::UseDisplayList (
            bool useList )  [virtual]
```

Enable or disable the use of Display Lists inside FTGL.

Parameters

useList	`true` turns ON display lists. `false` turns OFF display lists.

3.9.4 Friends And Related Symbol Documentation

3.9.4.1 FTBitmapFont

```
friend class FTBitmapFont  [friend]
```

Definition at line 78 of file FTFont.h.

3.9.4.2 FTBufferFont

```
friend class FTBufferFont  [friend]
```

Definition at line 79 of file FTFont.h.

3.9.4.3 FTExtrudeFont

```
friend class FTExtrudeFont  [friend]
```

Definition at line 80 of file FTFont.h.

3.9.4.4 FTFontImpl

```
friend class FTFontImpl  [friend]
```

Definition at line 382 of file FTFont.h.

3.9.4.5 FTOutlineFont

```
friend class FTOutlineFont  [friend]
```

Definition at line 81 of file FTFont.h.

3.9.4.6 FTPixmapFont

```
friend class FTPixmapFont  [friend]
```

Definition at line 82 of file FTFont.h.

3.9.4.7 FTPolygonFont

```
friend class FTPolygonFont  [friend]
```

Definition at line 83 of file FTFont.h.

3.9.4.8 FTTextureFont

```
friend class FTTextureFont  [friend]
```

Definition at line 84 of file FTFont.h.

3.9.4.9 FTTriangleExtractorFont

```
friend class FTTriangleExtractorFont  [friend]
```

Definition at line 85 of file FTFont.h.

The documentation for this class was generated from the following file:

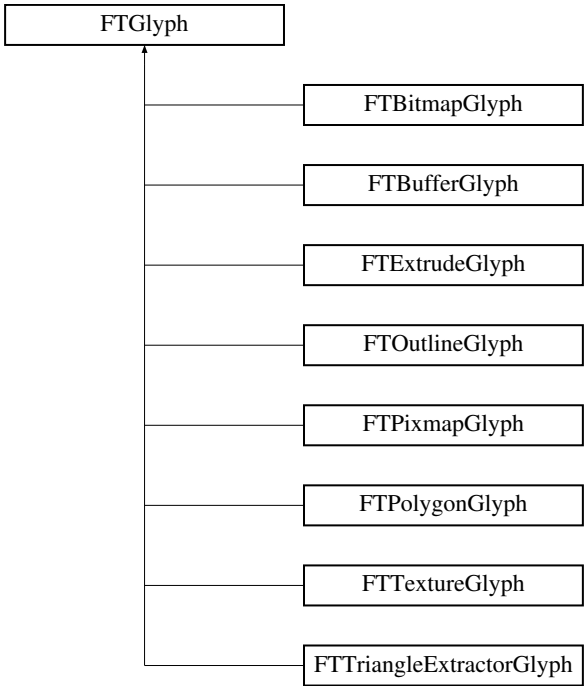
- FTFont.h

3.10 FTGlyph Class Reference

FTGlyph is the base class for FTGL glyphs.

`#include <FTGlyph.h>`

Inheritance diagram for FTGlyph:



Public Member Functions

- virtual ~FTGlyph ()

  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)=0

  Renders this glyph at the current pen position.
- virtual float Advance () const

  Return the advance width for this glyph.
- virtual const FTBBox & BBox () const

  Return the bounding box for this glyph.
- virtual FT_Error Error () const

  Queries for errors.

Protected Member Functions

- FTGlyph (FT_GlyphSlot glyph)

  Create a glyph.

Friends

- class FTBitmapGlyph
- class FTBufferGlyph
- class FTExtrudeGlyph
- class FTOutlineGlyph
- class FTPixmapGlyph
- class FTPolygonGlyph
- class FTTextureGlyph
- class FTTriangleExtractorGlyph

3.10.1 Detailed Description

FTGlyph is the base class for FTGL glyphs.

It provides the interface between Freetype glyphs and their openGL renderable counterparts. This is an abstract class and derived classes must implement the `Render` function.

See also

FTBBox

FTPoint

Definition at line 50 of file FTGlyph.h.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 FTGlyph()

```
FTGlyph::FTGlyph (
            FT_GlyphSlot glyph )  [protected]
```

Create a glyph.

Parameters

glyph	The Freetype glyph to be processed

3.10.2.2 ~FTGlyph()

```
virtual FTGlyph::~FTGlyph ( )  [virtual]
```

Destructor.

3.10.3 Member Function Documentation

3.10.3.1 Advance()

```
virtual float FTGlyph::Advance ( ) const  [virtual]
```

Return the advance width for this glyph.

Returns

advance width.

3.10.3.2 BBox()

```
virtual const FTBBox & FTGlyph::BBox ( ) const  [virtual]
```

Return the bounding box for this glyph.

Returns

bounding box.

3.10.3.3 Error()

```
virtual FT_Error FTGlyph::Error ( ) const  [virtual]
```

Queries for errors.

Returns

The current error code.

3.10.3.4 Render()

```
virtual const FTPoint & FTGlyph::Render (
             const FTPoint & pen,
             int renderMode )  [pure virtual]
```

Renders this glyph at the current pen position.

Parameters

pen	The current pen position.
renderMode	Render mode to display

Returns

The advance distance for this glyph.

Implemented in FTBitmapGlyph, FTBufferGlyph, FTExtrudeGlyph, FTOutlineGlyph, FTPixmapGlyph, FTPolygonGlyph, FTTextureGlyph, and FTTriangleExtractorGlyph.

3.10.4 Friends And Related Symbol Documentation

3.10.4.1 FTBitmapGlyph

```
friend class FTBitmapGlyph  [friend]
```

Definition at line 70 of file FTGlyph.h.

3.10.4.2 FTBufferGlyph

```
friend class FTBufferGlyph  [friend]
```

Definition at line 71 of file FTGlyph.h.

3.10.4.3 FTExtrudeGlyph

```
friend class FTExtrudeGlyph  [friend]
```

Definition at line 72 of file FTGlyph.h.

3.10.4.4 FTOutlineGlyph

```
friend class FTOutlineGlyph  [friend]
```

Definition at line 73 of file FTGlyph.h.

3.10.4.5 FTPixmapGlyph

```
friend class FTPixmapGlyph  [friend]
```

Definition at line 74 of file FTGlyph.h.

3.10.4.6 FTPolygonGlyph

```
friend class FTPolygonGlyph  [friend]
```

Definition at line 75 of file FTGlyph.h.

3.10.4.7 FTTextureGlyph

```
friend class FTTextureGlyph  [friend]
```

Definition at line 76 of file FTGlyph.h.

3.10.4.8 FTTriangleExtractorGlyph

```
friend class FTTriangleExtractorGlyph  [friend]
```

Definition at line 77 of file FTGlyph.h.

The documentation for this class was generated from the following file:

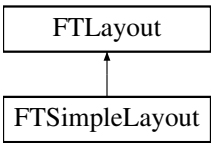
- FTGlyph.h

3.11 FTLayout Class Reference

FTLayout is the interface for layout managers that render text.

```
#include <FTLayout.h>
```

Inheritance diagram for FTLayout:



Public Member Functions

- virtual ~FTLayout ()

  Destructor.
- virtual FTBBox BBox (const char *string, const int len=-1, FTPoint position=FTPoint())=0

  Get the bounding box for a formatted string.
- virtual FTBBox BBox (const wchar_t *string, const int len=-1, FTPoint position=FTPoint())=0

  Get the bounding box for a formatted string.
- virtual void Render (const char *string, const int len=-1, FTPoint position=FTPoint(), int renderMode=FTGL::RENDER_ALL)=0

  Render a string of characters.
- virtual void Render (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), int renderMode=FTGL::RENDER_ALL)=0

  Render a string of characters.
- virtual FT_Error Error () const

  Queries the Layout for errors.

Protected Member Functions

- FTLayout ()

Friends

- class FTSimpleLayout

3.11.1 Detailed Description

FTLayout is the interface for layout managers that render text.

Specific layout manager classes are derived from this class. This class is abstract and deriving classes must implement the protected `Render` methods to render formatted text and `BBox` methods to determine the bounding box of output text.

See also

FTFont

FTBBox

Definition at line 52 of file FTLayout.h.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 FTLayout()

```
FTLayout::FTLayout ( )  [protected]
```

3.11.2.2 ~FTLayout()

```
virtual FTLayout::~FTLayout ( )  [virtual]
```

Destructor.

3.11.3 Member Function Documentation

3.11.3.1 BBox() [1/2]

```
virtual FTBBox FTLayout::BBox (
            const char * string,
            const int len = -1,
            FTPoint position = FTPoint() )  [pure virtual]
```

Get the bounding box for a formatted string.

Parameters

string	A char string.
len	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
position	The pen position of the first character (optional).

Returns

The corresponding bounding box.

Implemented in FTSimpleLayout.

3.11.3.2 BBox() [2/2]

```
virtual FTBBox FTLayout::BBox (
            const wchar_t * string,
            const int len = -1,
            FTPoint position = FTPoint() )  [pure virtual]
```

Get the bounding box for a formatted string.

Parameters

string	A wchar_t string.
len	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
position	The pen position of the first character (optional).

Returns

The corresponding bounding box.

Implemented in FTSimpleLayout.

3.11.3.3 Error()

```
virtual FT_Error FTLayout::Error ( ) const  [virtual]
```

Queries the Layout for errors.

Returns

The current error code.

3.11.3.4 Render() [1/2]

```
virtual void FTLayout::Render (
            const char * string,
            const int len = -1,
            FTPoint position = FTPoint(),
            int renderMode = FTGL::RENDER_ALL )  [pure virtual]
```

Render a string of characters.

Parameters

string	'C' style string to be output.
len	The length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).
position	The pen position of the first character (optional).
renderMode	Render mode to display (optional)

Implemented in FTSimpleLayout.

3.11.3.5 Render() [2/2]

```
virtual void FTLayout::Render (
             const wchar_t * string,
             const int len = -1,
             FTPoint position = FTPoint(),
             int renderMode = FTGL::RENDER_ALL )  [pure virtual]
```

Render a string of characters.

Parameters

string	wchar_t string to be output.
len	The length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).
position	The pen position of the first character (optional).
renderMode	Render mode to display (optional)

Implemented in FTSimpleLayout.

3.11.4 Friends And Related Symbol Documentation

3.11.4.1 FTSimpleLayout

```
friend class FTSimpleLayout  [friend]
```

Definition at line 67 of file FTLayout.h.

The documentation for this class was generated from the following file:

- FTLayout.h

3.12 FTLibrary Class Reference

FTLibrary class is the global accessor for the Freetype library.

```
#include <FTLibrary.h>
```

Public Member Functions

- const FT_Library * GetLibrary () const

  Gets a pointer to the native Freetype library.
- FT_Error Error () const

  Queries the library for errors.
- ~FTLibrary ()

  Destructor.
- void LegacyOpenGLState (bool On)

  See README-LegacyOpenGLState.
- bool GetLegacyOpenGLStateSet () const

Static Public Member Functions

- static FTLibrary & Instance ()

  Global acces point to the single FTLibrary object.

3.12.1 Detailed Description

FTLibrary class is the global accessor for the Freetype library.

This class encapsulates the Freetype Library. This is a singleton class and ensures that only one FT_Library is in existence at any one time. All constructors are private therefore clients cannot create or instantiate this class themselves and must access it's methods via the static `FTLibrary::Instance()` function.

Just because this class returns a valid `FTLibrary` object doesn't mean that the Freetype Library has been successfully initialised. Clients should check for errors. You can initialise the library AND check for errors using the following code... `err = FTLibrary::Instance().Error();`

See also

¨Freetype 2 Documentation¨

Definition at line 56 of file FTLibrary.h.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 ~FTLibrary()

```
FTLibrary::~FTLibrary ( )
```

Destructor.

Disposes of the Freetype library

3.12.3 Member Function Documentation

3.12.3.1 Error()

```
FT_Error FTLibrary::Error ( ) const  [inline]
```

Queries the library for errors.

Returns

The current error code.

Definition at line 78 of file FTLibrary.h.

3.12.3.2 GetLegacyOpenGLStateSet()

```
bool FTLibrary::GetLegacyOpenGLStateSet ( ) const  [inline]
```

Definition at line 97 of file FTLibrary.h.

3.12.3.3 GetLibrary()

```
const FT_Library * FTLibrary::GetLibrary ( ) const  [inline]
```

Gets a pointer to the native Freetype library.

Returns

A handle to a FreeType library instance.

Definition at line 71 of file FTLibrary.h.

3.12.3.4 Instance()

```
static FTLibrary & FTLibrary::Instance ( )  [static]
```

Global acces point to the single FTLibrary object.

Returns

The global `FTLibrary` object.

3.12.3.5 LegacyOpenGLState()

```
void FTLibrary::LegacyOpenGLState (
            bool On )
```

See README-LegacyOpenGLState.

Choose incompatible legacy behaviour, see commit 29603ae3fa88c5b9e079a6db23be2cdea95aef39.

May only be set to the same value (but any number of times) within one program.

The documentation for this class was generated from the following file:

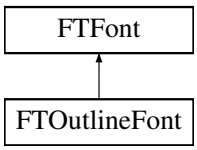
- FTLibrary.h

3.13 FTOutlineFont Class Reference

FTOutlineFont is a specialisation of the FTFont class for handling Vector Outline fonts.

```
#include <FTGLOutlineFont.h>
```

Inheritance diagram for FTOutlineFont:



Public Member Functions

- FTOutlineFont (const char *fontFilePath)

  Open and read a font file.
- FTOutlineFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.
- ~FTOutlineFont ()

  Destructor.

Public Member Functions inherited from FTFont

- virtual ~FTFont ()
- virtual bool Attach (const char *fontFilePath)

  Attach auxilliary file to font e.g font metrics.
- virtual bool Attach (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Attach auxilliary data to font e.g font metrics, from memory.
- virtual void GlyphLoadFlags (FT_Int flags)

  Set the glyph loading flags.
- virtual bool CharMap (FT_Encoding encoding)

  Set the character map for the face.
- virtual unsigned int CharMapCount () const

  Get the number of character maps in this face.
- virtual FT_Encoding * CharMapList ()

  Get a list of character maps in this face.
- virtual bool FaceSize (const unsigned int size, const unsigned int res=72)

  Set the char size for the current face.
- virtual unsigned int FaceSize () const

  Get the current face size in points (1/72 inch).
- virtual void Depth (float depth)

  Set the extrusion distance for the font.
- virtual void Outset (float outset)

  Set the outset distance for the font.
- virtual void Outset (float front, float back)

  Set the front and back outset distances for the font.
- virtual void UseDisplayList (bool useList)

  Enable or disable the use of Display Lists inside FTGL.
- virtual float Ascender () const

  Get the global ascender height for the face.
- virtual float Descender () const

  Gets the global descender height for the face.
- virtual float LineHeight () const

  Gets the line spacing for the font.
- virtual FTBBox BBox (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const char *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).
- virtual FTBBox BBox (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const wchar_t *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).
- virtual float Advance (const char *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual float Advance (const wchar_t *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual FTPoint Render (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FTPoint Render (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FT_Error Error () const

  Queries the Font for errors.

Protected Member Functions

- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)

  Construct a glyph of the correct type.

Protected Member Functions inherited from FTFont

- FTFont (char const *fontFilePath)

  Open and read a font file.
- FTFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.
- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)=0

  Construct a glyph of the correct type.

3.13.1 Detailed Description

FTOutlineFont is a specialisation of the FTFont class for handling Vector Outline fonts.

See also

FTFont

Definition at line 45 of file FTGLOutlineFont.h.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 FTOutlineFont() [1/2]

```
FTOutlineFont::FTOutlineFont (
            const char * fontFilePath )
```

Open and read a font file.

Sets Error flag.

Parameters

fontFilePath	font file path.

3.13.2.2 FTOutlineFont() [2/2]

```
FTOutlineFont::FTOutlineFont (
            const unsigned char * pBufferBytes,
            size_t bufferSizeInBytes )
```

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

pBufferBytes	the in-memory buffer
bufferSizeInBytes	the length of the buffer in bytes

3.13.2.3 ~FTOutlineFont()

```
FTOutlineFont::~FTOutlineFont ( )
```

Destructor.

3.13.3 Member Function Documentation

3.13.3.1 MakeGlyph()

```
virtual FTGlyph * FTOutlineFont::MakeGlyph (
            FT_GlyphSlot slot )  [protected], [virtual]
```

Construct a glyph of the correct type.

Clients must override the function and return their specialised FTGlyph.

Parameters

slot	A FreeType glyph slot.

Returns

An FT****Glyph or `null` on failure.

Implements FTFont.

The documentation for this class was generated from the following file:

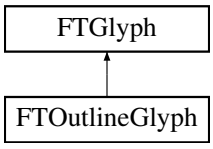
- FTGLOutlineFont.h

3.14 FTOutlineGlyph Class Reference

FTOutlineGlyph is a specialisation of FTGlyph for creating outlines.

```
#include <FTOutlineGlyph.h>
```

Inheritance diagram for FTOutlineGlyph:



Public Member Functions

- FTOutlineGlyph (FT_GlyphSlot glyph, float outset, bool useDisplayList)

  Constructor.
- virtual ~FTOutlineGlyph ()

  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)

  Render this glyph at the current pen position.

Public Member Functions inherited from FTGlyph

- virtual ~FTGlyph ()

  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)=0

  Renders this glyph at the current pen position.
- virtual float Advance () const

  Return the advance width for this glyph.
- virtual const FTBBox & BBox () const

  Return the bounding box for this glyph.
- virtual FT_Error Error () const

  Queries for errors.

Additional Inherited Members

Protected Member Functions inherited from FTGlyph

- FTGlyph (FT_GlyphSlot glyph)

  Create a glyph.

3.14.1 Detailed Description

FTOutlineGlyph is a specialisation of FTGlyph for creating outlines.

Definition at line 42 of file FTOutlineGlyph.h.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 FTOutlineGlyph()

```
FTOutlineGlyph::FTOutlineGlyph (
            FT_GlyphSlot glyph,
            float outset,
            bool useDisplayList )
```

Constructor.

Sets the Error to Invalid_Outline if the glyphs isn't an outline.

Parameters

glyph	The Freetype glyph to be processed
outset	outset distance
useDisplayList	Enable or disable the use of Display Lists for this glyph `true` turns ON display lists. `false` turns OFF display lists.

3.14.2.2 ~FTOutlineGlyph()

```
virtual FTOutlineGlyph::~FTOutlineGlyph ( )  [virtual]
```

Destructor.

3.14.3 Member Function Documentation

3.14.3.1 Render()

```
virtual const FTPoint & FTOutlineGlyph::Render (
             const FTPoint & pen,
             int renderMode )  [virtual]
```

Render this glyph at the current pen position.

Parameters

pen	The current pen position.
renderMode	Render mode to display

Returns

The advance distance for this glyph.

Implements FTGlyph.

The documentation for this class was generated from the following file:

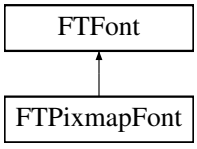
- FTOutlineGlyph.h

3.15 FTPixmapFont Class Reference

FTPixmapFont is a specialisation of the FTFont class for handling Pixmap (Grey Scale) fonts.

`#include <FTGLPixmapFont.h>`

Inheritance diagram for FTPixmapFont:



Public Member Functions

- FTPixmapFont (const char *fontFilePath)

  Open and read a font file.
- FTPixmapFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.
- ~FTPixmapFont ()

  Destructor.

Public Member Functions inherited from FTFont

- virtual ~FTFont ()
- virtual bool Attach (const char *fontFilePath)

  Attach auxilliary file to font e.g font metrics.
- virtual bool Attach (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Attach auxilliary data to font e.g font metrics, from memory.
- virtual void GlyphLoadFlags (FT_Int flags)

  Set the glyph loading flags.
- virtual bool CharMap (FT_Encoding encoding)

  Set the character map for the face.
- virtual unsigned int CharMapCount () const

  Get the number of character maps in this face.
- virtual FT_Encoding * CharMapList ()

  Get a list of character maps in this face.
- virtual bool FaceSize (const unsigned int size, const unsigned int res=72)

  Set the char size for the current face.
- virtual unsigned int FaceSize () const

  Get the current face size in points (1/72 inch).
- virtual void Depth (float depth)

  Set the extrusion distance for the font.
- virtual void Outset (float outset)

  Set the outset distance for the font.
- virtual void Outset (float front, float back)

  Set the front and back outset distances for the font.
- virtual void UseDisplayList (bool useList)

  Enable or disable the use of Display Lists inside FTGL.
- virtual float Ascender () const

  Get the global ascender height for the face.
- virtual float Descender () const

  Gets the global descender height for the face.
- virtual float LineHeight () const

  Gets the line spacing for the font.
- virtual FTBBox BBox (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const char *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).
- virtual FTBBox BBox (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const wchar_t *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

Get the bounding box for a string (deprecated).

- virtual float Advance (const char *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual float Advance (const wchar_t *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual FTPoint Render (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FTPoint Render (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FT_Error Error () const

  Queries the Font for errors.

Protected Member Functions

- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)

  Construct a glyph of the correct type.

Protected Member Functions inherited from FTFont

- FTFont (char const *fontFilePath)

  Open and read a font file.
- FTFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.
- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)=0

  Construct a glyph of the correct type.

3.15.1 Detailed Description

FTPixmapFont is a specialisation of the FTFont class for handling Pixmap (Grey Scale) fonts.

See also

FTFont

Definition at line 45 of file FTGLPixmapFont.h.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 FTPixmapFont() [1/2]

```
FTPixmapFont::FTPixmapFont (
            const char * fontFilePath )
```

Open and read a font file.

Sets Error flag.

Parameters

fontFilePath	font file path.

3.15.2.2 FTPixmapFont() [2/2]

```
FTPixmapFont::FTPixmapFont (
            const unsigned char * pBufferBytes,
            size_t bufferSizeInBytes )
```

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

pBufferBytes	the in-memory buffer
bufferSizeInBytes	the length of the buffer in bytes

3.15.2.3 ~FTPixmapFont()

```
FTPixmapFont::~FTPixmapFont ( )
```

Destructor.

3.15.3 Member Function Documentation

3.15.3.1 MakeGlyph()

```
virtual FTGlyph * FTPixmapFont::MakeGlyph (
            FT_GlyphSlot slot )  [protected], [virtual]
```

Construct a glyph of the correct type.

Clients must override the function and return their specialised FTGlyph.

Parameters

slot	A FreeType glyph slot.

Returns

An FT****Glyph or `null` on failure.

Implements FTFont.

The documentation for this class was generated from the following file:

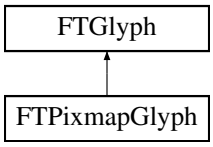
- FTGLPixmapFont.h

3.16 FTPixmapGlyph Class Reference

FTPixmapGlyph is a specialisation of FTGlyph for creating pixmaps.

`#include <FTPixmapGlyph.h>`

Inheritance diagram for FTPixmapGlyph:



Public Member Functions

- FTPixmapGlyph (FT_GlyphSlot glyph)

  Constructor.
- virtual ~FTPixmapGlyph ()

  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)

  Render this glyph at the current pen position.

Public Member Functions inherited from FTGlyph

- virtual ~FTGlyph ()

  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)=0

  Renders this glyph at the current pen position.
- virtual float Advance () const

  Return the advance width for this glyph.
- virtual const FTBBox & BBox () const

  Return the bounding box for this glyph.
- virtual FT_Error Error () const

  Queries for errors.

Additional Inherited Members

Protected Member Functions inherited from FTGlyph

- FTGlyph (FT_GlyphSlot glyph)

  Create a glyph.

3.16.1 Detailed Description

FTPixmapGlyph is a specialisation of FTGlyph for creating pixmaps.

Definition at line 42 of file FTPixmapGlyph.h.

3.16.2 Constructor & Destructor Documentation

3.16.2.1 FTPixmapGlyph()

```
FTPixmapGlyph::FTPixmapGlyph (
            FT_GlyphSlot glyph )
```

Constructor.

Parameters

glyph	The Freetype glyph to be processed

3.16.2.2 ~FTPixmapGlyph()

```
virtual FTPixmapGlyph::~FTPixmapGlyph ( )  [virtual]
```

Destructor.

3.16.3 Member Function Documentation

3.16.3.1 Render()

```
virtual const FTPoint & FTPixmapGlyph::Render (
            const FTPoint & pen,
            int renderMode )  [virtual]
```

Render this glyph at the current pen position.

Parameters

pen	The current pen position.
renderMode	Render mode to display

Returns

The advance distance for this glyph.

Implements FTGlyph.

The documentation for this class was generated from the following file:

- FTPixmapGlyph.h

3.17 FTPoint Class Reference

FTPoint class is a basic 3-dimensional point or vector.

`#include <FTPoint.h>`

Public Member Functions

- FTPoint ()
  Default constructor.
- FTPoint (const FTGL_DOUBLE x, const FTGL_DOUBLE y, const FTGL_DOUBLE z=0)
  Constructor.
- FTPoint (const FT_Vector &ft_vector)
  Constructor.
- FTPoint Normalise ()
  Normalise a point's coordinates.
- FTPoint & operator+= (const FTPoint &point)
  Operator += In Place Addition.
- FTPoint operator+ (const FTPoint &point) const
  Operator +.
- FTPoint & operator-= (const FTPoint &point)
  Operator -= In Place Substraction.
- FTPoint operator- (const FTPoint &point) const
  Operator -.
- FTPoint operator∗ (double multiplier) const
  Operator ∗ Scalar multiplication.
- FTPoint operator^ (const FTPoint &point)
  Operator ^ Vector product.
- operator const FTGL_DOUBLE ∗ () const
  Cast to FTGL_DOUBLE∗.
- void X (FTGL_DOUBLE x)
  Setters.
- void Y (FTGL_DOUBLE y)
- void Z (FTGL_DOUBLE z)
- FTGL_DOUBLE X () const
  Getters.
- FTGL_DOUBLE Y () const
- FTGL_DOUBLE Z () const
- FTGL_FLOAT Xf () const
- FTGL_FLOAT Yf () const
- FTGL_FLOAT Zf () const

Friends

- FTPoint operator∗ (double multiplier, FTPoint &point)
  Operator ∗ Scalar multiplication.
- double operator∗ (FTPoint &a, FTPoint &b)
  Operator ∗ Scalar product.
- bool operator== (const FTPoint &a, const FTPoint &b)
  Operator == Tests for equality.
- bool operator!= (const FTPoint &a, const FTPoint &b)
  Operator != Tests for non equality.

3.17.1 Detailed Description

FTPoint class is a basic 3-dimensional point or vector.

Definition at line 42 of file FTPoint.h.

3.17.2 Constructor & Destructor Documentation

3.17.2.1 FTPoint() [1/3]

```
FTPoint::FTPoint ( )  [inline]
```

Default constructor.

Point is set to zero.

Definition at line 48 of file FTPoint.h.

3.17.2.2 FTPoint() [2/3]

```
FTPoint::FTPoint (
            const FTGL_DOUBLE x,
            const FTGL_DOUBLE y,
            const FTGL_DOUBLE z = 0 )  [inline]
```

Constructor.

Z coordinate is set to zero if unspecified.

Parameters

x	First component
y	Second component
z	Third component

Definition at line 62 of file FTPoint.h.

3.17.2.3 FTPoint() [3/3]

```
FTPoint::FTPoint (
            const FT_Vector & ft_vector )  [inline]
```

Constructor.

This converts an FT_Vector to an FTPoint

Parameters

ft_vector	A freetype vector

Definition at line 75 of file FTPoint.h.

3.17.3 Member Function Documentation

3.17.3.1 Normalise()

```
FTPoint FTPoint::Normalise ( )
```

Normalise a point's coordinates.

If the coordinates are zero, the point is left untouched.

Returns

A vector of norm one.

3.17.3.2 operator const FTGL_DOUBLE *()

```
FTPoint::operator const FTGL_DOUBLE * ( ) const  [inline]
```

Cast to FTGL_DOUBLE*.

Definition at line 240 of file FTPoint.h.

3.17.3.3 operator*()

```
FTPoint FTPoint::operator* (
            double multiplier ) const  [inline]
```

Operator * Scalar multiplication.

Parameters

multiplier	

Returns

`this` multiplied by `multiplier`.

Definition at line 159 of file FTPoint.h.

3.17.3.4 operator+()

```
FTPoint FTPoint::operator+ (
            const FTPoint & point ) const  [inline]
```

Operator +.

Parameters

point	

Returns

this plus point.

Definition at line 112 of file FTPoint.h.

3.17.3.5 operator+=()

```
FTPoint & FTPoint::operator+= (
            const FTPoint & point )  [inline]
```

Operator += In Place Addition.

Parameters

point	

Returns

this plus point.

Definition at line 97 of file FTPoint.h.

3.17.3.6 operator-()

```
FTPoint FTPoint::operator- (
            const FTPoint & point ) const  [inline]
```

Operator -.

Parameters

point	

Returns

this minus point.

Definition at line 143 of file FTPoint.h.

3.17.3.7 operator-=()

```
FTPoint & FTPoint::operator-= (
            const FTPoint & point )  [inline]
```

Operator -= In Place Substraction.

Parameters

point	

Returns

this minus point.

Definition at line 128 of file FTPoint.h.

3.17.3.8 operator^()

```
FTPoint FTPoint::operator^ (
            const FTPoint & point )  [inline]
```

Operator ^ Vector product.

Parameters

point	Second point

Returns

this vector point.

Definition at line 204 of file FTPoint.h.

3.17.3.9 X() [1/2]

```
FTGL_DOUBLE FTPoint::X ( ) const  [inline]
```

Getters.

Definition at line 257 of file FTPoint.h.

3.17.3.10 X() [2/2]

```
void FTPoint::X (
            FTGL_DOUBLE x )  [inline]
```

Setters.

Definition at line 249 of file FTPoint.h.

Referenced by FTBBox::operator|=().

3.17.3.11 Xf()

```
FTGL_FLOAT FTPoint::Xf ( ) const  [inline]
```

Definition at line 260 of file FTPoint.h.

Referenced by FTFont::BBox(), and FTFont::BBox().

3.17.3.12 Y() [1/2]

```
FTGL_DOUBLE FTPoint::Y ( ) const  [inline]
```

Definition at line 258 of file FTPoint.h.

3.17.3.13 Y() [2/2]

```
void FTPoint::Y (
            FTGL_DOUBLE y )  [inline]
```

Definition at line 250 of file FTPoint.h.

Referenced by FTBBox::operator|=().

3.17.3.14 Yf()

```
FTGL_FLOAT FTPoint::Yf ( ) const  [inline]
```

Definition at line 261 of file FTPoint.h.

Referenced by FTFont::BBox(), and FTFont::BBox().

3.17.3.15 Z() [1/2]

```
FTGL_DOUBLE FTPoint::Z ( ) const  [inline]
```

Definition at line 259 of file FTPoint.h.

3.17.3.16 Z() [2/2]

```
void FTPoint::Z (
            FTGL_DOUBLE z )  [inline]
```

Definition at line 251 of file FTPoint.h.

Referenced by FTBBox::operator|=().

3.17.3.17 Zf()

```
FTGL_FLOAT FTPoint::Zf ( ) const  [inline]
```

Definition at line 262 of file FTPoint.h.

Referenced by FTFont::BBox(), and FTFont::BBox().

3.17.4 Friends And Related Symbol Documentation

3.17.4.1 operator"!=

```
bool operator!= (
            const FTPoint & a,
            const FTPoint & b )  [friend]
```

Operator != Tests for non equality.

Parameters

a	
b	

Returns

true if a & b are not equal

3.17.4.2 operator* [1/2]

```
FTPoint operator* (
            double multiplier,
            FTPoint & point )  [friend]
```

Operator * Scalar multiplication.

Parameters

point	
multiplier	

Returns

`multiplier` multiplied by `point`.

Definition at line 177 of file FTPoint.h.

3.17.4.3 operator* [2/2]

```
double operator* (
            FTPoint & a,
            FTPoint & b )  [friend]
```

Operator * Scalar product.

Parameters

a	First vector.
b	Second vector.

Returns

`a.b` scalar product.

Definition at line 190 of file FTPoint.h.

3.17.4.4 operator==

```
bool operator== (
            const FTPoint & a,
            const FTPoint & b )  [friend]
```

Operator == Tests for equality.

Parameters

a	
b	

Returns

true if a & b are equal

The documentation for this class was generated from the following file:

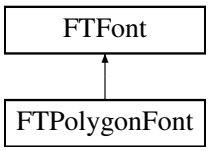
- FTPoint.h

3.18 FTPolygonFont Class Reference

FTPolygonFont is a specialisation of the FTFont class for handling tesselated Polygon Mesh fonts.

```
#include <FTGLPolygonFont.h>
```

Inheritance diagram for FTPolygonFont:



Public Member Functions

- FTPolygonFont (const char *fontFilePath)
  Open and read a font file.
- FTPolygonFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)
  Open and read a font from a buffer in memory.
- ~FTPolygonFont ()
  Destructor.

Public Member Functions inherited from FTFont

- virtual ~FTFont ()
- virtual bool Attach (const char *fontFilePath)

  Attach auxilliary file to font e.g font metrics.
- virtual bool Attach (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Attach auxilliary data to font e.g font metrics, from memory.
- virtual void GlyphLoadFlags (FT_Int flags)

  Set the glyph loading flags.
- virtual bool CharMap (FT_Encoding encoding)

  Set the character map for the face.
- virtual unsigned int CharMapCount () const

  Get the number of character maps in this face.
- virtual FT_Encoding * CharMapList ()

  Get a list of character maps in this face.
- virtual bool FaceSize (const unsigned int size, const unsigned int res=72)

  Set the char size for the current face.
- virtual unsigned int FaceSize () const

  Get the current face size in points (1/72 inch).
- virtual void Depth (float depth)

  Set the extrusion distance for the font.
- virtual void Outset (float outset)

  Set the outset distance for the font.
- virtual void Outset (float front, float back)

  Set the front and back outset distances for the font.
- virtual void UseDisplayList (bool useList)

  Enable or disable the use of Display Lists inside FTGL.
- virtual float Ascender () const

  Get the global ascender height for the face.
- virtual float Descender () const

  Gets the global descender height for the face.
- virtual float LineHeight () const

  Gets the line spacing for the font.
- virtual FTBBox BBox (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const char *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).
- virtual FTBBox BBox (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const wchar_t *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).
- virtual float Advance (const char *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual float Advance (const wchar_t *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual FTPoint Render (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FTPoint Render (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FT_Error Error () const

  Queries the Font for errors.

Protected Member Functions

- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)

  Construct a glyph of the correct type.

Protected Member Functions inherited from FTFont

- FTFont (char const *fontFilePath)

  Open and read a font file.
- FTFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.
- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)=0

  Construct a glyph of the correct type.

3.18.1 Detailed Description

FTPolygonFont is a specialisation of the FTFont class for handling tesselated Polygon Mesh fonts.

See also

FTFont

Definition at line 46 of file FTGLPolygonFont.h.

3.18.2 Constructor & Destructor Documentation

3.18.2.1 FTPolygonFont() [1/2]

```
FTPolygonFont::FTPolygonFont (
            const char * fontFilePath )
```

Open and read a font file.

Sets Error flag.

Parameters

fontFilePath	font file path.

3.18.2.2 FTPolygonFont() [2/2]

```
FTPolygonFont::FTPolygonFont (
            const unsigned char * pBufferBytes,
            size_t bufferSizeInBytes )
```

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

pBufferBytes	the in-memory buffer
bufferSizeInBytes	the length of the buffer in bytes

3.18.2.3 ~FTPolygonFont()

```
FTPolygonFont::~FTPolygonFont ( )
```

Destructor.

3.18.3 Member Function Documentation

3.18.3.1 MakeGlyph()

```
virtual FTGlyph * FTPolygonFont::MakeGlyph (
            FT_GlyphSlot slot )  [protected], [virtual]
```

Construct a glyph of the correct type.

Clients must override the function and return their specialised FTGlyph.

Parameters

slot	A FreeType glyph slot.

Returns

An FT****Glyph or `null` on failure.

Implements FTFont.

The documentation for this class was generated from the following file:

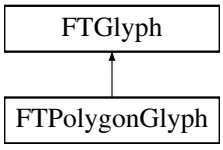
- FTGLPolygonFont.h

3.19 FTPolygonGlyph Class Reference

FTPolygonGlyph is a specialisation of FTGlyph for creating tessellated polygon glyphs.

```
#include <FTPolyGlyph.h>
```

Inheritance diagram for FTPolygonGlyph:



Public Member Functions

- FTPolygonGlyph (FT_GlyphSlot glyph, float outset, bool useDisplayList)

  Constructor.
- virtual ~FTPolygonGlyph ()

  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)

  Render this glyph at the current pen position.

Public Member Functions inherited from FTGlyph

- virtual ~FTGlyph ()

  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)=0

  Renders this glyph at the current pen position.
- virtual float Advance () const

  Return the advance width for this glyph.
- virtual const FTBBox & BBox () const

  Return the bounding box for this glyph.
- virtual FT_Error Error () const

  Queries for errors.

Additional Inherited Members

Protected Member Functions inherited from FTGlyph

- FTGlyph (FT_GlyphSlot glyph)

  Create a glyph.

3.19.1 Detailed Description

FTPolygonGlyph is a specialisation of FTGlyph for creating tessellated polygon glyphs.

Definition at line 43 of file FTPolyGlyph.h.

3.19.2 Constructor & Destructor Documentation

3.19.2.1 FTPolygonGlyph()

```
FTPolygonGlyph::FTPolygonGlyph (
            FT_GlyphSlot glyph,
            float outset,
            bool useDisplayList )
```

Constructor.

Sets the Error to Invalid_Outline if the glyphs isn't an outline.

Parameters

glyph	The Freetype glyph to be processed
outset	The outset distance
useDisplayList	Enable or disable the use of Display Lists for this glyph `true` turns ON display lists. `false` turns OFF display lists.

3.19.2.2 ~FTPolygonGlyph()

```
virtual FTPolygonGlyph::~FTPolygonGlyph ( )  [virtual]
```

Destructor.

3.19.3 Member Function Documentation

3.19.3.1 Render()

```
virtual const FTPoint & FTPolygonGlyph::Render (
             const FTPoint & pen,
             int renderMode )  [virtual]
```

Render this glyph at the current pen position.

Parameters

pen	The current pen position.
renderMode	Render mode to display

Returns

The advance distance for this glyph.

Implements FTGlyph.

The documentation for this class was generated from the following file:

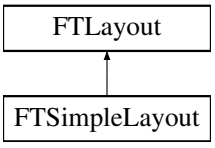
- FTPolyGlyph.h

3.20 FTSimpleLayout Class Reference

FTSimpleLayout is a specialisation of FTLayout for simple text boxes.

`#include <FTSimpleLayout.h>`

Inheritance diagram for FTSimpleLayout:



Public Member Functions

- FTSimpleLayout ()

  Initializes line spacing to 1.0, alignment to ALIGN_LEFT and wrap to 100.0.
- ~FTSimpleLayout ()

  Destructor.
- virtual FTBBox BBox (const char *string, const int len=-1, FTPoint position=FTPoint())

  Get the bounding box for a formatted string.
- virtual FTBBox BBox (const wchar_t *string, const int len=-1, FTPoint position=FTPoint())

  Get the bounding box for a formatted string.
- virtual void Render (const char *string, const int len=-1, FTPoint position=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual void Render (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- void SetFont (FTFont *fontInit)

  Set the font to use for rendering the text.
- FTFont * GetFont ()
- void SetLineLength (const float LineLength)

  The maximum line length for formatting text.
- float GetLineLength () const
- void SetAlignment (const FTGL::TextAlignment Alignment)

  The text alignment mode used to distribute space within a line or rendered text.
- FTGL::TextAlignment GetAlignment () const
- void SetLineSpacing (const float LineSpacing)

  Sets the line height.
- float GetLineSpacing () const

Public Member Functions inherited from FTLayout

- virtual ~FTLayout ()

  Destructor.
- virtual FTBBox BBox (const char *string, const int len=-1, FTPoint position=FTPoint())=0

  Get the bounding box for a formatted string.
- virtual FTBBox BBox (const wchar_t *string, const int len=-1, FTPoint position=FTPoint())=0

  Get the bounding box for a formatted string.
- virtual void Render (const char *string, const int len=-1, FTPoint position=FTPoint(), int renderMode=FTGL::RENDER_ALL)=0

  Render a string of characters.
- virtual void Render (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), int renderMode=FTGL::RENDER_ALL)=0

  Render a string of characters.
- virtual FT_Error Error () const

  Queries the Layout for errors.

Additional Inherited Members

Protected Member Functions inherited from FTLayout

- FTLayout ()

3.20.1 Detailed Description

FTSimpleLayout is a specialisation of FTLayout for simple text boxes.

This class has basic support for text wrapping, left, right and centered alignment, and text justification.

See also

FTLayout

Definition at line 49 of file FTSimpleLayout.h.

3.20.2 Constructor & Destructor Documentation

3.20.2.1 FTSimpleLayout()

```
FTSimpleLayout::FTSimpleLayout ( )
```

Initializes line spacing to 1.0, alignment to ALIGN_LEFT and wrap to 100.0.

3.20.2.2 ~FTSimpleLayout()

```
FTSimpleLayout::~FTSimpleLayout ( )
```

Destructor.

3.20.3 Member Function Documentation

3.20.3.1 BBox() [1/2]

```
virtual FTBBox FTSimpleLayout::BBox (
            const char * string,
            const int len = -1,
            FTPoint position = FTPoint() )  [virtual]
```

Get the bounding box for a formatted string.

Parameters

string	A char string.
len	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
position	The pen position of the first character (optional).

Returns

The corresponding bounding box.

Implements FTLayout.

3.20.3.2 BBox() [2/2]

```
virtual FTBBox FTSimpleLayout::BBox (
            const wchar_t * string,
            const int len = -1,
            FTPoint position = FTPoint() )  [virtual]
```

Get the bounding box for a formatted string.

Parameters

string	A wchar_t string.
len	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
position	The pen position of the first character (optional).

Returns

The corresponding bounding box.

Implements FTLayout.

3.20.3.3 GetAlignment()

```
FTGL::TextAlignment FTSimpleLayout::GetAlignment ( ) const
```

Returns

The text alignment mode.

3.20.3.4 GetFont()

```
FTFont * FTSimpleLayout::GetFont ( )
```

Returns

The current font.

3.20.3.5 GetLineLength()

```
float FTSimpleLayout::GetLineLength ( ) const
```

Returns

The current line length.

3.20.3.6 GetLineSpacing()

```
float FTSimpleLayout::GetLineSpacing ( ) const
```

Returns

The line spacing.

3.20.3.7 Render() [1/2]

```
virtual void FTSimpleLayout::Render (
            const char * string,
            const int len = -1,
            FTPoint position = FTPoint(),
            int renderMode = FTGL::RENDER_ALL )  [virtual]
```

Render a string of characters.

Parameters

string	'C' style string to be output.
len	The length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).
position	The pen position of the first character (optional).
renderMode	Render mode to display (optional)

Implements FTLayout.

3.20.3.8 Render() [2/2]

```
virtual void FTSimpleLayout::Render (
            const wchar_t * string,
            const int len = -1,
            FTPoint position = FTPoint(),
            int renderMode = FTGL::RENDER_ALL )  [virtual]
```

Render a string of characters.

Parameters

string	wchar_t string to be output.
len	The length of the string. If < 0 then all characters will be displayed until a null character is encountered (optional).
position	The pen position of the first character (optional).
renderMode	Render mode to display (optional)

Implements FTLayout.

3.20.3.9 SetAlignment()

```
void FTSimpleLayout::SetAlignment (
            const FTGL::TextAlignment Alignment )
```

The text alignment mode used to distribute space within a line or rendered text.

Parameters

Alignment	The new alignment mode.

3.20.3.10 SetFont()

```
void FTSimpleLayout::SetFont (
            FTFont * fontInit )
```

Set the font to use for rendering the text.

Parameters

fontInit	A pointer to the new font. The font is referenced by this but will not be disposed of when this is deleted.

3.20.3.11 SetLineLength()

```
void FTSimpleLayout::SetLineLength (
            const float LineLength )
```

The maximum line length for formatting text.

Parameters

LineLength	The new line length.

3.20.3.12 SetLineSpacing()

```
void FTSimpleLayout::SetLineSpacing (
            const float LineSpacing )
```

Sets the line height.

Parameters

LineSpacing	The height of each line of text expressed as a percentage of the current fonts line height.

The documentation for this class was generated from the following file:

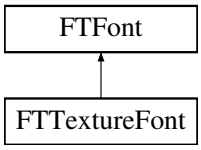
- FTSimpleLayout.h

3.21 FTTextureFont Class Reference

FTTextureFont is a specialisation of the FTFont class for handling Texture mapped fonts.

`#include <FTGLTextureFont.h>`

Inheritance diagram for FTTextureFont:



Public Member Functions

- FTTextureFont (const char *fontFilePath)

  Open and read a font file.
- FTTextureFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.
- virtual ~FTTextureFont ()

  Destructor.

Public Member Functions inherited from FTFont

- virtual ~FTFont ()
- virtual bool Attach (const char *fontFilePath)

  Attach auxilliary file to font e.g font metrics.
- virtual bool Attach (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Attach auxilliary data to font e.g font metrics, from memory.
- virtual void GlyphLoadFlags (FT_Int flags)

  Set the glyph loading flags.
- virtual bool CharMap (FT_Encoding encoding)

  Set the character map for the face.
- virtual unsigned int CharMapCount () const

  Get the number of character maps in this face.
- virtual FT_Encoding * CharMapList ()

  Get a list of character maps in this face.
- virtual bool FaceSize (const unsigned int size, const unsigned int res=72)

  Set the char size for the current face.
- virtual unsigned int FaceSize () const

  Get the current face size in points (1/72 inch).
- virtual void Depth (float depth)

  Set the extrusion distance for the font.
- virtual void Outset (float outset)

  Set the outset distance for the font.
- virtual void Outset (float front, float back)

  Set the front and back outset distances for the font.
- virtual void UseDisplayList (bool useList)

  Enable or disable the use of Display Lists inside FTGL.

- virtual float Ascender () const

  Get the global ascender height for the face.
- virtual float Descender () const

  Gets the global descender height for the face.
- virtual float LineHeight () const

  Gets the line spacing for the font.
- virtual FTBBox BBox (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const char *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).
- virtual FTBBox BBox (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const wchar_t *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).
- virtual float Advance (const char *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual float Advance (const wchar_t *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual FTPoint Render (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FTPoint Render (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FT_Error Error () const

  Queries the Font for errors.

Protected Member Functions

- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)

  Construct a glyph of the correct type.

Protected Member Functions inherited from FTFont

- FTFont (char const *fontFilePath)

  Open and read a font file.
- FTFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.
- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)=0

  Construct a glyph of the correct type.

3.21.1 Detailed Description

FTTextureFont is a specialisation of the FTFont class for handling Texture mapped fonts.

See also

FTFont

Definition at line 45 of file FTGLTextureFont.h.

3.21.2 Constructor & Destructor Documentation

3.21.2.1 FTTextureFont() [1/2]

```
FTTextureFont::FTTextureFont (
            const char * fontFilePath )
```

Open and read a font file.

Sets Error flag.

Parameters

fontFilePath	font file path.

3.21.2.2 FTTextureFont() [2/2]

```
FTTextureFont::FTTextureFont (
            const unsigned char * pBufferBytes,
            size_t bufferSizeInBytes )
```

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

pBufferBytes	the in-memory buffer
bufferSizeInBytes	the length of the buffer in bytes

3.21.2.3 ~FTTextureFont()

```
virtual FTTextureFont::~FTTextureFont ( )  [virtual]
```

Destructor.

3.21.3 Member Function Documentation

3.21.3.1 MakeGlyph()

```
virtual FTGlyph * FTTextureFont::MakeGlyph (
            FT_GlyphSlot slot )  [protected], [virtual]
```

Construct a glyph of the correct type.

Clients must override the function and return their specialised FTGlyph.

Parameters

slot	A FreeType glyph slot.

Returns

An FT****Glyph or `null` on failure.

Implements FTFont.

The documentation for this class was generated from the following file:

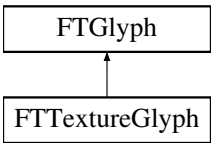
- FTGLTextureFont.h

3.22 FTTextureGlyph Class Reference

FTTextureGlyph is a specialisation of FTGlyph for creating texture glyphs.

`#include <FTTextureGlyph.h>`

Inheritance diagram for FTTextureGlyph:



Public Member Functions

- FTTextureGlyph (FT_GlyphSlot glyph, int id, int xOffset, int yOffset, int width, int height)
  Constructor.
- virtual ~FTTextureGlyph ()
  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)
  Render this glyph at the current pen position.

Public Member Functions inherited from FTGlyph

- virtual ~FTGlyph ()
  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)=0
  Renders this glyph at the current pen position.
- virtual float Advance () const
  Return the advance width for this glyph.
- virtual const FTBBox & BBox () const
  Return the bounding box for this glyph.
- virtual FT_Error Error () const
  Queries for errors.

Additional Inherited Members

Protected Member Functions inherited from FTGlyph

- FTGlyph (FT_GlyphSlot glyph)
  Create a glyph.

3.22.1 Detailed Description

FTTextureGlyph is a specialisation of FTGlyph for creating texture glyphs.

Definition at line 43 of file FTTextureGlyph.h.

3.22.2 Constructor & Destructor Documentation

3.22.2.1 FTTextureGlyph()

```
FTTextureGlyph::FTTextureGlyph (
            FT_GlyphSlot glyph,
            int id,
            int xOffset,
            int yOffset,
            int width,
            int height )
```

Constructor.

Parameters

glyph	The Freetype glyph to be processed
id	The id of the texture that this glyph will be drawn in
xOffset	The x offset into the parent texture to draw this glyph
yOffset	The y offset into the parent texture to draw this glyph
width	The width of the parent texture
height	The height (number of rows) of the parent texture

3.22.2.2 ~FTTextureGlyph()

```
virtual FTTextureGlyph::~FTTextureGlyph ( )  [virtual]
```

Destructor.

3.22.3 Member Function Documentation

3.22.3.1 Render()

```
virtual const FTPoint & FTTextureGlyph::Render (
            const FTPoint & pen,
            int renderMode )  [virtual]
```

Render this glyph at the current pen position.

Parameters

pen	The current pen position.
renderMode	Render mode to display

Returns

The advance distance for this glyph.

Implements FTGlyph.

The documentation for this class was generated from the following file:

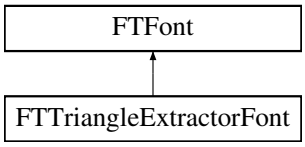
- FTTextureGlyph.h

3.23 FTTriangleExtractorFont Class Reference

FTPolygonFont is a specialisation of the FTFont class for handling tesselated Polygon Mesh fonts.

`#include <FTGLTriangleExtractorFont.h>`

Inheritance diagram for FTTriangleExtractorFont:



Public Member Functions

- FTTriangleExtractorFont (const char ∗fontFilePath, std::vector< float > &triangles)

  Open and read a font file.
- FTTriangleExtractorFont (const unsigned char ∗pBufferBytes, size_t bufferSizeInBytes, std::vector< float > &triangles)

  Open and read a font from a buffer in memory.
- ∼FTTriangleExtractorFont ()

  Destructor.

Public Member Functions inherited from FTFont

- virtual ~FTFont ()
- virtual bool Attach (const char *fontFilePath)

  Attach auxilliary file to font e.g font metrics.
- virtual bool Attach (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Attach auxilliary data to font e.g font metrics, from memory.
- virtual void GlyphLoadFlags (FT_Int flags)

  Set the glyph loading flags.
- virtual bool CharMap (FT_Encoding encoding)

  Set the character map for the face.
- virtual unsigned int CharMapCount () const

  Get the number of character maps in this face.
- virtual FT_Encoding * CharMapList ()

  Get a list of character maps in this face.
- virtual bool FaceSize (const unsigned int size, const unsigned int res=72)

  Set the char size for the current face.
- virtual unsigned int FaceSize () const

  Get the current face size in points (1/72 inch).
- virtual void Depth (float depth)

  Set the extrusion distance for the font.
- virtual void Outset (float outset)

  Set the outset distance for the font.
- virtual void Outset (float front, float back)

  Set the front and back outset distances for the font.
- virtual void UseDisplayList (bool useList)

  Enable or disable the use of Display Lists inside FTGL.
- virtual float Ascender () const

  Get the global ascender height for the face.
- virtual float Descender () const

  Gets the global descender height for the face.
- virtual float LineHeight () const

  Gets the line spacing for the font.
- virtual FTBBox BBox (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const char *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).
- virtual FTBBox BBox (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint())

  Get the bounding box for a string.
- void BBox (const wchar_t *string, float &llx, float &lly, float &llz, float &urx, float &ury, float &urz)

  Get the bounding box for a string (deprecated).
- virtual float Advance (const char *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual float Advance (const wchar_t *string, const int len=-1, FTPoint spacing=FTPoint())

  Get the advance for a string.
- virtual FTPoint Render (const char *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FTPoint Render (const wchar_t *string, const int len=-1, FTPoint position=FTPoint(), FTPoint spacing=FTPoint(), int renderMode=FTGL::RENDER_ALL)

  Render a string of characters.
- virtual FT_Error Error () const

  Queries the Font for errors.

Protected Member Functions

- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)

  Construct a glyph of the correct type.

Protected Member Functions inherited from FTFont

- FTFont (char const *fontFilePath)

  Open and read a font file.
- FTFont (const unsigned char *pBufferBytes, size_t bufferSizeInBytes)

  Open and read a font from a buffer in memory.
- virtual FTGlyph * MakeGlyph (FT_GlyphSlot slot)=0

  Construct a glyph of the correct type.

3.23.1 Detailed Description

FTPolygonFont is a specialisation of the FTFont class for handling tesselated Polygon Mesh fonts.

See also

FTFont

Definition at line 44 of file FTGLTriangleExtractorFont.h.

3.23.2 Constructor & Destructor Documentation

3.23.2.1 FTTriangleExtractorFont() [1/2]

```
FTTriangleExtractorFont::FTTriangleExtractorFont (
            const char * fontFilePath,
            std::vector< float > & triangles )
```

Open and read a font file.

Sets Error flag.

Parameters

fontFilePath	font file path.
triangles	the container to store the triangle data.

3.23.2.2 FTTriangleExtractorFont() [2/2]

```
FTTriangleExtractorFont::FTTriangleExtractorFont (
            const unsigned char * pBufferBytes,
```

```
            size_t bufferSizeInBytes,
            std::vector< float > & triangles )
```

Open and read a font from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

pBufferBytes	the in-memory buffer
bufferSizeInBytes	the length of the buffer in bytes
triangles	the container to store the triangle data.

3.23.2.3 ~FTTriangleExtractorFont()

```
FTTriangleExtractorFont::~FTTriangleExtractorFont ( )
```

Destructor.

3.23.3 Member Function Documentation

3.23.3.1 MakeGlyph()

```
virtual FTGlyph * FTTriangleExtractorFont::MakeGlyph (
            FT_GlyphSlot slot )  [protected], [virtual]
```

Construct a glyph of the correct type.

Clients must override the function and return their specialised FTGlyph.

Parameters

slot	A FreeType glyph slot.

Returns

An FT****Glyph or `null` on failure.

Implements FTFont.

The documentation for this class was generated from the following file:

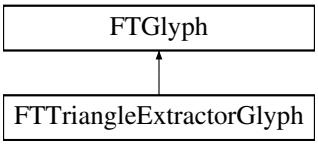
- FTGLTriangleExtractorFont.h

3.24 FTTriangleExtractorGlyph Class Reference

FTPolygonGlyph is a specialisation of FTGlyph for creating tessellated polygon glyphs.

`#include <FTTriangleExtractorGlyph.h>`

Inheritance diagram for FTTriangleExtractorGlyph:



Public Member Functions

- FTTriangleExtractorGlyph (FT_GlyphSlot glyph, float outset, std::vector< float > &triangles)
  Constructor.
- virtual ~FTTriangleExtractorGlyph ()
  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)
  Render this glyph at the current pen position.

Public Member Functions inherited from FTGlyph

- virtual ~FTGlyph ()
  Destructor.
- virtual const FTPoint & Render (const FTPoint &pen, int renderMode)=0
  Renders this glyph at the current pen position.
- virtual float Advance () const
  Return the advance width for this glyph.
- virtual const FTBBox & BBox () const
  Return the bounding box for this glyph.
- virtual FT_Error Error () const
  Queries for errors.

Additional Inherited Members

Protected Member Functions inherited from FTGlyph

- FTGlyph (FT_GlyphSlot glyph)
  Create a glyph.

3.24.1 Detailed Description

FTPolygonGlyph is a specialisation of FTGlyph for creating tessellated polygon glyphs.

Definition at line 42 of file FTTriangleExtractorGlyph.h.

3.24.2 Constructor & Destructor Documentation

3.24.2.1 FTTriangleExtractorGlyph()

```
FTTriangleExtractorGlyph::FTTriangleExtractorGlyph (
            FT_GlyphSlot glyph,
            float outset,
            std::vector< float > & triangles )
```

Constructor.

Sets the Error to Invalid_Outline if the glyphs isn't an outline.

Parameters

glyph	The Freetype glyph to be processed
outset	The outset distance
useDisplayList	Enable or disable the use of Display Lists for this glyph `true` turns ON display lists. `false` turns OFF display lists.

3.24.2.2 ~FTTriangleExtractorGlyph()

```
virtual FTTriangleExtractorGlyph::~FTTriangleExtractorGlyph ( )  [virtual]
```

Destructor.

3.24.3 Member Function Documentation

3.24.3.1 Render()

```
virtual const FTPoint & FTTriangleExtractorGlyph::Render (
            const FTPoint & pen,
            int renderMode )  [virtual]
```

Render this glyph at the current pen position.

Parameters

pen	The current pen position.
renderMode	Render mode to display

Returns

The advance distance for this glyph.

Implements FTGlyph.

The documentation for this class was generated from the following file:

- FTTriangleExtractorGlyph.h

Chapter 4

File Documentation

4.1 faq.dox File Reference

4.2 ftgl.dox File Reference

4.3 projects_using_ftgl.txt File Reference

4.4 tutorial.dox File Reference

4.5 FTBBox.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTBBox

  FTBBox is a convenience class for handling bounding boxes.

4.6 FTBBox.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
```

```
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTBBox__
00034 #define __FTBBox__
00035
00036 #ifdef __cplusplus
00037
00038
00042 class FTGL_EXPORT FTBBox
00043 {
00044     public:
00048         FTBBox()
00049         :   lower(0.0f, 0.0f, 0.0f),
00050             upper(0.0f, 0.0f, 0.0f)
00051         {}
00052
00056         FTBBox(float lx, float ly, float lz, float ux, float uy, float uz)
00057         :   lower(lx, ly, lz),
00058             upper(ux, uy, uz)
00059         {}
00060
00064         FTBBox(FTPoint l, FTPoint u)
00065         :   lower(l),
00066             upper(u)
00067         {}
00068
00075         FTBBox(FT_GlyphSlot glyph)
00076         :   lower(0.0f, 0.0f, 0.0f),
00077             upper(0.0f, 0.0f, 0.0f)
00078         {
00079             FT_BBox bbox;
00080             FT_Outline_Get_CBox(&(glyph->outline), &bbox);
00081
00082             lower.X(static_cast<float>(bbox.xMin) / 64.0f);
00083             lower.Y(static_cast<float>(bbox.yMin) / 64.0f);
00084             lower.Z(0.0f);
00085             upper.X(static_cast<float>(bbox.xMax) / 64.0f);
00086             upper.Y(static_cast<float>(bbox.yMax) / 64.0f);
00087             upper.Z(0.0f);
00088         }
00089
00093         ~FTBBox()
00094         {}
00095
00100         void Invalidate()
00101         {
00102             lower = FTPoint(1.0f, 1.0f, 1.0f);
00103             upper = FTPoint(-1.0f, -1.0f, -1.0f);
00104         }
00105
00112         bool IsValid()
00113         {
00114             return lower.X() <= upper.X()
00115                 && lower.Y() <= upper.Y()
00116                 && lower.Z() <= upper.Z();
00117         }
00118
00124         FTBBox& operator += (const FTPoint vector)
00125         {
00126             lower += vector;
00127             upper += vector;
00128
00129             return *this;
00130         }
00131
00138         FTBBox& operator |= (const FTBBox& bbox)
00139         {
00140             if(bbox.lower.X() < lower.X()) lower.X(bbox.lower.X());
00141             if(bbox.lower.Y() < lower.Y()) lower.Y(bbox.lower.Y());
00142             if(bbox.lower.Z() < lower.Z()) lower.Z(bbox.lower.Z());
```

```
00143             if(bbox.upper.X() > upper.X()) upper.X(bbox.upper.X());
00144             if(bbox.upper.Y() > upper.Y()) upper.Y(bbox.upper.Y());
00145             if(bbox.upper.Z() > upper.Z()) upper.Z(bbox.upper.Z());
00146
00147             return *this;
00148         }
00149
00150         void SetDepth(float depth)
00151         {
00152             if(depth > 0)
00153                 upper.Z(lower.Z() + depth);
00154             else
00155                 lower.Z(upper.Z() + depth);
00156         }
00157
00158
00159         inline FTPoint const Upper() const
00160         {
00161             return upper;
00162         }
00163
00164
00165         inline FTPoint const Lower() const
00166         {
00167             return lower;
00168         }
00169
00170     private:
00174         FTPoint lower, upper;
00175 };
00176
00177 #endif //__cplusplus
00178
00179 #endif  //  __FTBBox__
00180
```

4.7 FTBitmapGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTBitmapGlyph

  FTBitmapGlyph is a specialisation of FTGlyph for creating bitmaps.

Functions

- FTGLglyph * ftglCreateBitmapGlyph (FT_GlyphSlot glyph)

  Create a specialisation of FTGLglyph for creating bitmaps.

4.7.1 Function Documentation

4.7.1.1 ftglCreateBitmapGlyph()

```
FTGLglyph * ftglCreateBitmapGlyph (
            FT_GlyphSlot glyph )
```

Create a specialisation of FTGLglyph for creating bitmaps.

Parameters

glyph	The Freetype glyph to be processed

Returns

An FTGLglyph* object.

4.8 FTBitmapGlyph.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTBitmapGlyph__
00034 #define __FTBitmapGlyph__
00035
00036 #ifdef __cplusplus
00037
00038
00042 class FTGL_EXPORT FTBitmapGlyph : public FTGlyph
00043 {
00044     public:
00050         FTBitmapGlyph(FT_GlyphSlot glyph);
00051
00055         virtual ~FTBitmapGlyph();
00056
00064         virtual const FTPoint& Render(const FTPoint& pen, int renderMode);
00065 };
00066
00067 #endif //__cplusplus
00068
00069 FTGL_BEGIN_C_DECLS
00070
00077 FTGL_EXPORT FTGLglyph *ftglCreateBitmapGlyph(FT_GlyphSlot glyph);
00078
00079 FTGL_END_C_DECLS
00080
00081 #endif  //  __FTBitmapGlyph__
00082
```

4.9 FTBuffer.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTBuffer

  FTBuffer is a helper class for pixel buffers.

4.10 FTBuffer.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00005  *
00006  * Permission is hereby granted, free of charge, to any person obtaining
00007  * a copy of this software and associated documentation files (the
00008  * "Software"), to deal in the Software without restriction, including
00009  * without limitation the rights to use, copy, modify, merge, publish,
00010  * distribute, sublicense, and/or sell copies of the Software, and to
00011  * permit persons to whom the Software is furnished to do so, subject to
00012  * the following conditions:
00013  *
00014  * The above copyright notice and this permission notice shall be
00015  * included in all copies or substantial portions of the Software.
00016  *
00017  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00018  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00019  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00020  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00021  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00022  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00023  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00024  */
00025
00026 #ifndef __ftgl__
00027 #   warning Please use <FTGL/ftgl.h> instead of <FTBuffer.h>.
00028 #   include <FTGL/ftgl.h>
00029 #endif
00030
00031 #ifndef __FTBuffer__
00032 #define __FTBuffer__
00033
00034 #ifdef __cplusplus
00035
00045 class FTGL_EXPORT FTBuffer
00046 {
00047     public:
00051         FTBuffer();
00052
00056         ~FTBuffer();
00057
00063         inline FTPoint Pos() const
00064         {
00065             return pos;
00066         }
00067
00073         inline void Pos(FTPoint arg)
00074         {
00075             pos = arg;
00076         }
00077
00084         void Size(int w, int h);
00085
00091         inline int Width() const { return width; }
00092
00098         inline int Height() const { return height; }
00099
00105         inline unsigned char *Pixels() const { return pixels; }
00106
00107     private:
00111         int width, height;
00112
00116         unsigned char *pixels;
00117
00121         FTPoint pos;
00122 };
00123
00124 #endif //__cplusplus
00125
00126 #endif // __FTBuffer__
00127
```

4.11 FTBufferFont.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTBufferFont

  FTBufferFont is a specialisation of the FTFont class for handling memory buffer fonts.

Functions

- FTGLfont * ftglCreateBufferFont (const char *file)

  Create a specialised FTGLfont object for handling memory buffer fonts.

4.11.1 Function Documentation

4.11.1.1 ftglCreateBufferFont()

```
FTGLfont * ftglCreateBufferFont (
            const char * file )
```

Create a specialised FTGLfont object for handling memory buffer fonts.

Parameters

file	The font file name.

Returns

An FTGLfont* object.

See also

FTGLfont

4.12 FTBufferFont.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00005  *
00006  * Permission is hereby granted, free of charge, to any person obtaining
00007  * a copy of this software and associated documentation files (the
00008  * "Software"), to deal in the Software without restriction, including
00009  * without limitation the rights to use, copy, modify, merge, publish,
00010  * distribute, sublicense, and/or sell copies of the Software, and to
00011  * permit persons to whom the Software is furnished to do so, subject to
```

```
00012  * the following conditions:
00013  *
00014  * The above copyright notice and this permission notice shall be
00015  * included in all copies or substantial portions of the Software.
00016  *
00017  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00018  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00019  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00020  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00021  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00022  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00023  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00024  */
00025
00026 #ifndef __ftgl__
00027 #   warning Please use <FTGL/ftgl.h> instead of <FTBufferFont.h>.
00028 #   include <FTGL/ftgl.h>
00029 #endif
00030
00031 #ifndef __FTBufferFont__
00032 #define __FTBufferFont__
00033
00034 #ifdef __cplusplus
00035
00036
00043 class FTGL_EXPORT FTBufferFont : public FTFont
00044 {
00045     public:
00051         FTBufferFont(const char* fontFilePath);
00052
00061         FTBufferFont(const unsigned char *pBufferBytes,
00062                      size_t bufferSizeInBytes);
00063
00067         ~FTBufferFont();
00068
00069     protected:
00079         virtual FTGlyph* MakeGlyph(FT_GlyphSlot slot);
00080 };
00081
00082 #endif //__cplusplus
00083
00084 FTGL_BEGIN_C_DECLS
00085
00094 FTGL_EXPORT FTGLfont *ftglCreateBufferFont(const char *file);
00095
00096 FTGL_END_C_DECLS
00097
00098 #endif  //  __FTBufferFont__
00099
```

4.13 FTBufferGlyph.h File Reference

`#include <FTGL/ftgl.h>`

Data Structures

- class FTBufferGlyph

  FTBufferGlyph is a specialisation of FTGlyph for memory buffer rendering.

4.14 FTBufferGlyph.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00005  *
00006  * Permission is hereby granted, free of charge, to any person obtaining
00007  * a copy of this software and associated documentation files (the
```

```
00008  * "Software"), to deal in the Software without restriction, including
00009  * without limitation the rights to use, copy, modify, merge, publish,
00010  * distribute, sublicense, and/or sell copies of the Software, and to
00011  * permit persons to whom the Software is furnished to do so, subject to
00012  * the following conditions:
00013  *
00014  * The above copyright notice and this permission notice shall be
00015  * included in all copies or substantial portions of the Software.
00016  *
00017  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00018  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00019  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00020  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00021  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00022  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00023  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00024  */
00025
00026 #ifndef __ftgl__
00027 #   warning Please use <FTGL/ftgl.h> instead of <FTBufferGlyph.h>.
00028 #   include <FTGL/ftgl.h>
00029 #endif
00030
00031 #ifndef __FTBufferGlyph__
00032 #define __FTBufferGlyph__
00033
00034 #ifdef __cplusplus
00035
00036
00040 class FTGL_EXPORT FTBufferGlyph : public FTGlyph
00041 {
00042     public:
00049         FTBufferGlyph(FT_GlyphSlot glyph, FTBuffer *buffer);
00050
00054         virtual ~FTBufferGlyph();
00055
00063         virtual const FTPoint& Render(const FTPoint& pen, int renderMode);
00064 };
00065
00066 #endif //__cplusplus
00067
00068 #endif  //  __FTBufferGlyph__
00069
```

4.15 FTExtrdGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTExtrudeGlyph

  FTExtrudeGlyph is a specialisation of FTGlyph for creating tessellated extruded polygon glyphs.

Macros

- #define FTExtrdGlyph FTExtrudeGlyph

Functions

- FTGLglyph * ftglCreateExtrudeGlyph (FT_GlyphSlot glyph, float depth, float frontOutset, float backOutset, int useDisplayList)

  Create a specialisation of FTGLglyph for creating tessellated extruded polygon glyphs.

4.15.1 Macro Definition Documentation

4.15.1.1 FTExtrdGlyph

```
#define FTExtrdGlyph FTExtrudeGlyph
```

Definition at line 77 of file FTExtrdGlyph.h.

4.15.2 Function Documentation

4.15.2.1 ftglCreateExtrudeGlyph()

```
FTGLglyph * ftglCreateExtrudeGlyph (
            FT_GlyphSlot glyph,
            float depth,
            float frontOutset,
            float backOutset,
            int useDisplayList )
```

Create a specialisation of FTGLglyph for creating tessellated extruded polygon glyphs.

Parameters

glyph	The Freetype glyph to be processed
depth	The distance along the z axis to extrude the glyph
frontOutset	outset contour size
backOutset	outset contour size
useDisplayList	Enable or disable the use of Display Lists for this glyph `true` turns ON display lists. `false` turns OFF display lists.

Returns

An FTGLglyph* object.

4.16 FTExtrdGlyph.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
```

```
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTExtrudeGlyph__
00034 #define __FTExtrudeGlyph__
00035
00036 #ifdef __cplusplus
00037
00038
00043 class FTGL_EXPORT FTExtrudeGlyph : public FTGlyph
00044 {
00045     public:
00059         FTExtrudeGlyph(FT_GlyphSlot glyph, float depth, float frontOutset,
00060                        float backOutset, bool useDisplayList);
00061
00065         virtual ~FTExtrudeGlyph();
00066
00074         virtual const FTPoint& Render(const FTPoint& pen, int renderMode);
00075 };
00076
00077 #define FTExtrdGlyph FTExtrudeGlyph
00078
00079 #endif //__cplusplus
00080
00081 FTGL_BEGIN_C_DECLS
00082
00097 FTGL_EXPORT FTGLglyph *ftglCreateExtrudeGlyph(FT_GlyphSlot glyph, float depth,
00098                                 float frontOutset, float backOutset,
00099                                 int useDisplayList);
00100
00101 FTGL_END_C_DECLS
00102
00103 #endif  //  __FTExtrudeGlyph__
00104
```

4.17 FTFont.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTFont

  FTFont is the public interface for the FTGL library.

Typedefs

- typedef struct _FTGLfont FTGLfont

Functions

- FTGLfont * ftglCreateCustomFont (char const *fontFilePath, void *data, FTGLglyph *(*makeglyph↩ Callback)(FT_GlyphSlot, void *))

  Create a custom FTGL font object.
- FTGLfont * ftglCreateCustomFontFromMem (const unsigned char *bytes, size_t len, void *data, FTGLglyph *(*makeglyphCallback)(FT_GlyphSlot, void *))

Create a custom FTGL font object from a buffer in memory.

- void ftglDestroyFont (FTGLfont *font)

  Destroy an FTGL font object.
- int ftglAttachFile (FTGLfont *font, const char *path)

  Attach auxilliary file to font e.g.
- int ftglAttachData (FTGLfont *font, const unsigned char *data, size_t size)

  Attach auxilliary data to font, e.g.
- void ftglSetFontGlyphLoadFlags (FTGLfont *font, FT_Int flags)

  Set the glyph loading flags.
- int ftglSetFontCharMap (FTGLfont *font, FT_Encoding encoding)

  Set the character map for the face.
- unsigned int ftglGetFontCharMapCount (FTGLfont *font)

  Get the number of character maps in this face.
- FT_Encoding * ftglGetFontCharMapList (FTGLfont *font)

  Get a list of character maps in this face.
- int ftglSetFontFaceSize (FTGLfont *font, unsigned int size, unsigned int res)

  Set the char size for the current face.
- unsigned int ftglGetFontFaceSize (FTGLfont *font)

  Get the current face size in points (1/72 inch).
- void ftglSetFontDepth (FTGLfont *font, float depth)

  Set the extrusion distance for the font.
- void ftglSetFontOutset (FTGLfont *font, float front, float back)

  Set the outset distance for the font.
- void ftglSetFontDisplayList (FTGLfont *font, int useList)

  Enable or disable the use of Display Lists inside FTGL.
- float ftglGetFontAscender (FTGLfont *font)

  Get the global ascender height for the face.
- float ftglGetFontDescender (FTGLfont *font)

  Gets the global descender height for the face.
- float ftglGetFontLineHeight (FTGLfont *font)

  Gets the line spacing for the font.
- void ftglGetFontBBox (FTGLfont *font, const char *string, int len, float bounds[6])

  Get the bounding box for a string.
- float ftglGetFontAdvance (FTGLfont *font, const char *string)

  Get the advance width for a string.
- void ftglRenderFont (FTGLfont *font, const char *string, int mode)

  Render a string of characters.
- FT_Error ftglGetFontError (FTGLfont *font)

  Query a font for errors.

4.17.1 Typedef Documentation

4.17.1.1 FTGLfont

```
typedef struct _FTGLfont FTGLfont
```

Definition at line 414 of file FTFont.h.

4.17.2 Function Documentation

4.17.2.1 ftglAttachData()

```
int ftglAttachData (
            FTGLfont * font,
            const unsigned char * data,
            size_t size )
```

Attach auxilliary data to font, e.g.

font metrics, from memory.

Note: not all font formats implement this function.

Parameters

font	An FTGLfont* object.
data	The in-memory buffer.
size	The length of the buffer in bytes.

Returns

1 if file has been attached successfully.

4.17.2.2 ftglAttachFile()

```
int ftglAttachFile (
            FTGLfont * font,
            const char * path )
```

Attach auxilliary file to font e.g.

font metrics.

Note: not all font formats implement this function.

Parameters

font	An FTGLfont* object.
path	Auxilliary font file path.

Returns

1 if file has been attached successfully.

4.17.2.3 ftglCreateCustomFont()

```
FTGLfont * ftglCreateCustomFont (
            char const * fontFilePath,
```

```
            void * data,
            FTGLglyph *(*)(FT_GlyphSlot, void *) makeglyphCallback )
```

Create a custom FTGL font object.

Parameters

fontFilePath	The font file name.
data	A pointer to private data that will be passed to callbacks.
makeglyphCallback	A glyph-making callback function.

Returns

An FTGLfont* object.

4.17.2.4 ftglCreateCustomFontFromMem()

```
FTGLfont * ftglCreateCustomFontFromMem (
            const unsigned char * bytes,
            size_t len,
            void * data,
            FTGLglyph *(*)(FT_GlyphSlot, void *) makeglyphCallback )
```

Create a custom FTGL font object from a buffer in memory.

Parameters

bytes	the in-memory buffer
len	the length of the buffer in bytes
data	A pointer to private data that will be passed to callbacks.
makeglyphCallback	A glyph-making callback function.

Returns

An FTGLfont* object.

4.17.2.5 ftglDestroyFont()

```
void ftglDestroyFont (
            FTGLfont * font )
```

Destroy an FTGL font object.

Parameters

font	An FTGLfont* object.

4.17.2.6 ftglGetFontAdvance()

```
float ftglGetFontAdvance (
            FTGLfont * font,
            const char * string )
```

Get the advance width for a string.

Parameters

font	An FTGLfont* object.
string	A char string.

Returns

Advance width

4.17.2.7 ftglGetFontAscender()

```
float ftglGetFontAscender (
            FTGLfont * font )
```

Get the global ascender height for the face.

Parameters

font	An FTGLfont* object.

Returns

Ascender height

4.17.2.8 ftglGetFontBBox()

```
void ftglGetFontBBox (
            FTGLfont * font,
            const char * string,
            int len,
            float bounds[6] )
```

Get the bounding box for a string.

Parameters

font	An FTGLfont* object.
string	A char buffer
len	The length of the string. If < 0 then all characters will be checked until a null character is encountered (optional).
bounds	An array of 6 float values where the bounding box's lower left near and upper right far 3D coordinates will be stored.

4.17.2.9 ftglGetFontCharMapCount()

```
unsigned int ftglGetFontCharMapCount (
            FTGLfont * font )
```

Get the number of character maps in this face.

Parameters

font	An FTGLfont* object.

Returns

character map count.

4.17.2.10 ftglGetFontCharMapList()

```
FT_Encoding * ftglGetFontCharMapList (
            FTGLfont * font )
```

Get a list of character maps in this face.

Parameters

font	An FTGLfont* object.

Returns

pointer to the first encoding.

4.17.2.11 ftglGetFontDescender()

```
float ftglGetFontDescender (
            FTGLfont * font )
```

Gets the global descender height for the face.

Parameters

font	An FTGLfont* object.

Returns

Descender height

4.17.2.12 ftglGetFontError()

```
FT_Error ftglGetFontError (
```

```
            FTGLfont * font )
```

Query a font for errors.

Parameters

font	An FTGLfont* object.

Returns

The current error code.

4.17.2.13 ftglGetFontFaceSize()

```
unsigned int ftglGetFontFaceSize (
            FTGLfont * font )
```

Get the current face size in points (1/72 inch).

Parameters

font	An FTGLfont* object.

Returns

face size

4.17.2.14 ftglGetFontLineHeight()

```
float ftglGetFontLineHeight (
            FTGLfont * font )
```

Gets the line spacing for the font.

Parameters

font	An FTGLfont* object.

Returns

Line height

4.17.2.15 ftglRenderFont()

```
void ftglRenderFont (
            FTGLfont * font,
```

```
            const char * string,
            int mode )
```

Render a string of characters.

Parameters

font	An FTGLfont* object.
string	Char string to be output.
mode	Render mode to display.

4.17.2.16 ftglSetFontCharMap()

```
int ftglSetFontCharMap (
            FTGLfont * font,
            FT_Encoding encoding )
```

Set the character map for the face.

Parameters

font	An FTGLfont* object.
encoding	Freetype enumerate for char map code.

Returns

1 if charmap was valid and set correctly.

4.17.2.17 ftglSetFontDepth()

```
void ftglSetFontDepth (
            FTGLfont * font,
            float depth )
```

Set the extrusion distance for the font.

Only implemented by FTExtrudeFont.

Parameters

font	An FTGLfont* object.
depth	The extrusion distance.

4.17.2.18 ftglSetFontDisplayList()

```
void ftglSetFontDisplayList (
            FTGLfont * font,
            int useList )
```

Enable or disable the use of Display Lists inside FTGL.

Parameters

font	An FTGLfont* object.
useList	1 turns ON display lists. 0 turns OFF display lists.

4.17.2.19 ftglSetFontFaceSize()

```
int ftglSetFontFaceSize (
            FTGLfont * font,
            unsigned int size,
            unsigned int res )
```

Set the char size for the current face.

Parameters

font	An FTGLfont* object.
size	The face size in points (1/72 inch).
res	The resolution of the target device, or 0 to use the default value of 72.

Returns

1 if size was set correctly.

4.17.2.20 ftglSetFontGlyphLoadFlags()

```
void ftglSetFontGlyphLoadFlags (
            FTGLfont * font,
            FT_Int flags )
```

Set the glyph loading flags.

By default, fonts use the most sensible flags when loading a font's glyph using FT_Load_Glyph(). This function allows to override the default flags.

Parameters

font	An FTGLfont* object.
flags	The glyph loading flags.

4.17.2.21 ftglSetFontOutset()

```
void ftglSetFontOutset (
            FTGLfont * font,
```

```
            float front,
            float back )
```

Set the outset distance for the font.

Only FTOutlineFont, FTPolygonFont and FTExtrudeFont implement front outset. Only FTExtrudeFont implements back outset.

Parameters

font	An FTGLfont* object.
front	The front outset distance.
back	The back outset distance.

4.18 FTFont.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTFont__
00034 #define __FTFont__
00035
00036 #ifdef __cplusplus
00037
00038 class FTFontImpl;
00039
00056 class FTGL_EXPORT FTFont
00057 {
00058     protected:
00064         FTFont(char const *fontFilePath);
00065
00074         FTFont(const unsigned char *pBufferBytes, size_t bufferSizeInBytes);
00075
00076     private:
00077         /* Allow our internal subclasses to access the private constructor */
00078         friend class FTBitmapFont;
00079         friend class FTBufferFont;
00080         friend class FTExtrudeFont;
00081         friend class FTOutlineFont;
00082         friend class FTPixmapFont;
00083         friend class FTPolygonFont;
00084         friend class FTTextureFont;
00085         friend class FTTriangleExtractorFont;
```

```
00086
00093         FTFont(FTFontImpl *pImpl);
00094
00095     public:
00096         virtual ~FTFont();
00097
00107         virtual bool Attach(const char* fontFilePath);
00108
00119         virtual bool Attach(const unsigned char *pBufferBytes,
00120                             size_t bufferSizeInBytes);
00121
00129         virtual void GlyphLoadFlags(FT_Int flags);
00130
00138         virtual bool CharMap(FT_Encoding encoding);
00139
00145         virtual unsigned int CharMapCount() const;
00146
00152         virtual FT_Encoding* CharMapList();
00153
00161         virtual bool FaceSize(const unsigned int size,
00162                               const unsigned int res = 72);
00163
00169         virtual unsigned int FaceSize() const;
00170
00177         virtual void Depth(float depth);
00178
00185         virtual void Outset(float outset);
00186
00194         virtual void Outset(float front, float back);
00195
00202         virtual void UseDisplayList(bool useList);
00203
00209         virtual float Ascender() const;
00210
00216         virtual float Descender() const;
00217
00223         virtual float LineHeight() const;
00224
00237         virtual FTBBox BBox(const char *string, const int len = -1,
00238                             FTPoint position = FTPoint(),
00239                             FTPoint spacing = FTPoint());
00240
00252         void BBox(const char* string, float& llx, float& lly, float& llz,
00253                   float& urx, float& ury, float& urz)
00254         {
00255             FTBBox b = BBox(string);
00256             llx = b.Lower().Xf(); lly = b.Lower().Yf(); llz = b.Lower().Zf();
00257             urx = b.Upper().Xf(); ury = b.Upper().Yf(); urz = b.Upper().Zf();
00258         }
00259
00272         virtual FTBBox BBox(const wchar_t *string, const int len = -1,
00273                             FTPoint position = FTPoint(),
00274                             FTPoint spacing = FTPoint());
00275
00287         void BBox(const wchar_t* string, float& llx, float& lly, float& llz,
00288                   float& urx, float& ury, float& urz)
00289         {
00290             FTBBox b = BBox(string);
00291             llx = b.Lower().Xf(); lly = b.Lower().Yf(); llz = b.Lower().Zf();
00292             urx = b.Upper().Xf(); ury = b.Upper().Yf(); urz = b.Upper().Zf();
00293         }
00294
00306         virtual float Advance(const char* string, const int len = -1,
00307                               FTPoint spacing = FTPoint());
00308
00320         virtual float Advance(const wchar_t* string, const int len = -1,
00321                               FTPoint spacing = FTPoint());
00322
00343         virtual FTPoint Render(const char* string, const int len = -1,
00344                                FTPoint position = FTPoint(),
00345                                FTPoint spacing = FTPoint(),
00346                                int renderMode = FTGL::RENDER_ALL);
00347
00368         virtual FTPoint Render(const wchar_t *string, const int len = -1,
00369                                FTPoint position = FTPoint(),
00370                                FTPoint spacing = FTPoint(),
00371                                int renderMode = FTGL::RENDER_ALL);
00372
00378         virtual FT_Error Error() const;
00379
00380     protected:
00381         /* Allow impl to access MakeGlyph */
00382         friend class FTFontImpl;
00383
00393         virtual FTGlyph* MakeGlyph(FT_GlyphSlot slot) = 0;
00394
00395     private:
```

```
00399         FTFontImpl *impl;
00400 };
00401
00402 #endif //__cplusplus
00403
00404 FTGL_BEGIN_C_DECLS
00405
00413 struct _FTGLFont;
00414 typedef struct _FTGLfont FTGLfont;
00415
00424 FTGL_EXPORT FTGLfont *ftglCreateCustomFont(char const *fontFilePath,
00425                                            void *data,
00426                     FTGLglyph * (*makeglyphCallback) (FT_GlyphSlot, void *));
00427
00437 FTGL_EXPORT FTGLfont *ftglCreateCustomFontFromMem(const unsigned char *bytes,
00438                                                   size_t len, void *data,
00439                     FTGLglyph * (*makeglyphCallback) (FT_GlyphSlot, void *));
00440
00446 FTGL_EXPORT void ftglDestroyFont(FTGLfont* font);
00447
00457 FTGL_EXPORT int ftglAttachFile(FTGLfont* font, const char* path);
00458
00469 FTGL_EXPORT int ftglAttachData(FTGLfont* font, const unsigned char * data,
00470                                size_t size);
00471
00480 FTGL_EXPORT void ftglSetFontGlyphLoadFlags(FTGLfont* font, FT_Int flags);
00481
00489 FTGL_EXPORT int ftglSetFontCharMap(FTGLfont* font, FT_Encoding encoding);
00490
00497 FTGL_EXPORT unsigned int ftglGetFontCharMapCount(FTGLfont* font);
00498
00505 FTGL_EXPORT FT_Encoding* ftglGetFontCharMapList(FTGLfont* font);
00506
00516 FTGL_EXPORT int ftglSetFontFaceSize(FTGLfont* font, unsigned int size,
00517                                     unsigned int res);
00518
00525 FTGL_EXPORT unsigned int ftglGetFontFaceSize(FTGLfont* font);
00526
00534 FTGL_EXPORT void ftglSetFontDepth(FTGLfont* font, float depth);
00535
00545 FTGL_EXPORT void ftglSetFontOutset(FTGLfont* font, float front, float back);
00546
00554 FTGL_EXPORT void ftglSetFontDisplayList(FTGLfont* font, int useList);
00555
00562 FTGL_EXPORT float ftglGetFontAscender(FTGLfont* font);
00563
00570 FTGL_EXPORT float ftglGetFontDescender(FTGLfont* font);
00571
00578 FTGL_EXPORT float ftglGetFontLineHeight(FTGLfont* font);
00579
00590 FTGL_EXPORT void ftglGetFontBBox(FTGLfont* font, const char *string,
00591                                  int len, float bounds[6]);
00592
00600 FTGL_EXPORT float ftglGetFontAdvance(FTGLfont* font, const char *string);
00601
00609 FTGL_EXPORT void ftglRenderFont(FTGLfont* font, const char *string, int mode);
00610
00617 FTGL_EXPORT FT_Error ftglGetFontError(FTGLfont* font);
00618
00619 FTGL_END_C_DECLS
00620
00621 #endif  //  __FTFont__
00622
```

4.19 ftgl.h File Reference

```
#include <ft2build.h>
#include <FT_FREETYPE_H>
#include <FT_GLYPH_H>
#include <FT_OUTLINE_H>
#include <FTGL/FTLibrary.h>
#include <FTGL/FTPoint.h>
#include <FTGL/FTBBox.h>
#include <FTGL/FTBuffer.h>
#include <FTGL/FTGlyph.h>
#include <FTGL/FTBitmapGlyph.h>
```

```
#include <FTGL/FTBufferGlyph.h>
#include <FTGL/FTExtrdGlyph.h>
#include <FTGL/FTOutlineGlyph.h>
#include <FTGL/FTPixmapGlyph.h>
#include <FTGL/FTPolyGlyph.h>
#include <FTGL/FTTextureGlyph.h>
#include <FTGL/FTTriangleExtractorGlyph.h>
#include <FTGL/FTFont.h>
#include <FTGL/FTGLBitmapFont.h>
#include <FTGL/FTBufferFont.h>
#include <FTGL/FTGLExtrdFont.h>
#include <FTGL/FTGLOutlineFont.h>
#include <FTGL/FTGLPixmapFont.h>
#include <FTGL/FTGLPolygonFont.h>
#include <FTGL/FTGLTextureFont.h>
#include <FTGL/FTGLTriangleExtractorFont.h>
#include <FTGL/FTLayout.h>
#include <FTGL/FTSimpleLayout.h>
```

Namespaces

- namespace FTGL

Macros

- #define FTGL_BEGIN_C_DECLS extern ¨C¨ { namespace FTGL {
- #define FTGL_END_C_DECLS } }
- #define FTGL_EXPORT

Typedefs

- typedef double FTGL_DOUBLE
- typedef float FTGL_FLOAT

Enumerations

- enum FTGL::RenderMode { FTGL::RENDER_FRONT = 0x0001 , FTGL::RENDER_BACK = 0x0002 , FTGL::RENDER_SIDE = 0x0004 , FTGL::RENDER_ALL = 0xffff }
- enum FTGL::TextAlignment { FTGL::ALIGN_LEFT = 0 , FTGL::ALIGN_CENTER = 1 , FTGL::ALIGN_RIGHT = 2 , FTGL::ALIGN_JUSTIFY = 3 }
- enum FTGL::ConfigString { FTGL::CONFIG_VERSION = 1 }

Functions

- char const * FTGL::GetString (ConfigString config)

  Return a string describing the current FTGL instance.

4.19.1 Macro Definition Documentation

4.19.1.1 FTGL_BEGIN_C_DECLS

```
#define FTGL_BEGIN_C_DECLS extern ¨C¨ { namespace FTGL {
```

Definition at line 43 of file ftgl.h.

4.19.1.2 FTGL_END_C_DECLS

```
#define FTGL_END_C_DECLS } }
```

Definition at line 44 of file ftgl.h.

4.19.1.3 FTGL_EXPORT

```
#define FTGL_EXPORT
```

Definition at line 134 of file ftgl.h.

4.19.2 Typedef Documentation

4.19.2.1 FTGL_DOUBLE

```
typedef double FTGL_DOUBLE
```

Definition at line 38 of file ftgl.h.

4.19.2.2 FTGL_FLOAT

```
typedef float FTGL_FLOAT
```

Definition at line 39 of file ftgl.h.

4.20 ftgl.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #define __ftgl__
00030
00031 /* We need the Freetype headers */
00032 #include <ft2build.h>
00033 #include FT_FREETYPE_H
00034 #include FT_GLYPH_H
00035 #include FT_OUTLINE_H
00036
00037 /* Floating point types used by the library */
00038 typedef double   FTGL_DOUBLE;
00039 typedef float    FTGL_FLOAT;
00040
00041 /* Macros used to declare C-linkage types and symbols */
00042 #ifdef __cplusplus
00043 #   define FTGL_BEGIN_C_DECLS extern "C" { namespace FTGL {
00044 #   define FTGL_END_C_DECLS } }
00045 #else
00046 #   define FTGL_BEGIN_C_DECLS
00047 #   define FTGL_END_C_DECLS
00048 #endif
00049
00050 #ifdef __cplusplus
00051 namespace FTGL
00052 {
00053     typedef enum
00054     {
00055         RENDER_FRONT = 0x0001,
00056         RENDER_BACK  = 0x0002,
00057         RENDER_SIDE  = 0x0004,
00058         RENDER_ALL   = 0xffff
00059     } RenderMode;
00060
00061     typedef enum
00062     {
00063         ALIGN_LEFT    = 0,
00064         ALIGN_CENTER  = 1,
00065         ALIGN_RIGHT   = 2,
00066         ALIGN_JUSTIFY = 3
00067     } TextAlignment;
00068
00069     typedef enum
00070     {
00071         CONFIG_VERSION = 1,
00072     } ConfigString;
00073
00082     extern char const *GetString(ConfigString config);
00083 }
00084 #else
00085 #   define FTGL_RENDER_FRONT 0x0001
00086 #   define FTGL_RENDER_BACK  0x0002
00087 #   define FTGL_RENDER_SIDE  0x0004
00088 #   define FTGL_RENDER_ALL   0xffff
00089
00090 #   define FTGL_ALIGN_LEFT    0
```

```
00091 #   define FTGL_ALIGN_CENTER  1
00092 #   define FTGL_ALIGN_RIGHT   2
00093 #   define FTGL_ALIGN_JUSTIFY 3
00094
00095 #   define FTGL_CONFIG_VERSION 1
00096
00105     extern char const *ftglGetString(int config);
00106 #endif
00107
00108 // Compiler-specific conditional compilation
00109 #ifdef _MSC_VER // MS Visual C++
00110
00111     // Disable various warning.
00112     // 4786: template name too long
00113     #pragma warning(disable : 4251)
00114     #pragma warning(disable : 4275)
00115     #pragma warning(disable : 4786)
00116
00117     // The following definitions control how symbols are exported.
00118     // If the target is a static library ensure that FTGL_LIBRARY_STATIC
00119     // is defined. If building a dynamic library (ie DLL) ensure the
00120     // FTGL_LIBRARY macro is defined, as it will mark symbols for
00121     // export. If compiling a project to _use_ the _dynamic_ library
00122     // version of the library, no definition is required.
00123     #ifdef FTGL_LIBRARY_STATIC      // static lib - no special export required
00124     #  define FTGL_EXPORT
00125     #elif FTGL_LIBRARY              // dynamic lib - must export/import symbols appropriately.
00126     #  define FTGL_EXPORT   __declspec(dllexport)
00127     #else
00128     #  define FTGL_EXPORT   __declspec(dllimport)
00129     #endif
00130
00131 #else
00132     // Compiler that is not MS Visual C++.
00133     // Ensure that the export symbol is defined (and blank)
00134     #define FTGL_EXPORT
00135 #endif
00136
00137 #include <FTGL/FTLibrary.h>
00138
00139 #include <FTGL/FTPoint.h>
00140 #include <FTGL/FTBBox.h>
00141 #include <FTGL/FTBuffer.h>
00142
00143 #include <FTGL/FTGlyph.h>
00144 #include <FTGL/FTBitmapGlyph.h>
00145 #include <FTGL/FTBufferGlyph.h>
00146 #include <FTGL/FTExtrdGlyph.h>
00147 #include <FTGL/FTOutlineGlyph.h>
00148 #include <FTGL/FTPixmapGlyph.h>
00149 #include <FTGL/FTPolyGlyph.h>
00150 #include <FTGL/FTTextureGlyph.h>
00151 #include <FTGL/FTTriangleExtractorGlyph.h>
00152
00153 #include <FTGL/FTFont.h>
00154 #include <FTGL/FTGLBitmapFont.h>
00155 #include <FTGL/FTBufferFont.h>
00156 #include <FTGL/FTGLExtrdFont.h>
00157 #include <FTGL/FTGLOutlineFont.h>
00158 #include <FTGL/FTGLPixmapFont.h>
00159 #include <FTGL/FTGLPolygonFont.h>
00160 #include <FTGL/FTGLTextureFont.h>
00161 #include <FTGL/FTGLTriangleExtractorFont.h>
00162
00163 #include <FTGL/FTLayout.h>
00164 #include <FTGL/FTSimpleLayout.h>
00165
00166 #endif  //  __ftgl__
```

4.21 FTGLBitmapFont.h File Reference

`#include <FTGL/ftgl.h>`

Data Structures

- class FTBitmapFont

  FTBitmapFont is a specialisation of the FTFont class for handling Bitmap fonts.

Macros

- #define FTGLBitmapFont FTBitmapFont

Functions

- FTGLfont * ftglCreateBitmapFont (const char *file)

  Create a specialised FTGLfont object for handling bitmap fonts.
- FTGLfont * ftglCreateBitmapFontFromMem (const unsigned char *bytes, size_t len)

  Create a specialised FTGLfont object for handling bitmap fonts from a buffer in memory.

4.21.1 Macro Definition Documentation

4.21.1.1 FTGLBitmapFont

```
#define FTGLBitmapFont FTBitmapFont
```

Definition at line 84 of file FTGLBitmapFont.h.

4.21.2 Function Documentation

4.21.2.1 ftglCreateBitmapFont()

```
FTGLfont * ftglCreateBitmapFont (
            const char * file )
```

Create a specialised FTGLfont object for handling bitmap fonts.

Parameters

file	The font file name.

Returns

An FTGLfont* object.

See also

FTGLfont

4.21.2.2 ftglCreateBitmapFontFromMem()

```
FTGLfont * ftglCreateBitmapFontFromMem (
            const unsigned char * bytes,
            size_t len )
```

Create a specialised FTGLfont object for handling bitmap fonts from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

bytes	the in-memory buffer
len	the length of the buffer in bytes

Returns

An FTGLfont* object.

4.22 FTGLBitmapFont.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTBitmapFont__
00034 #define __FTBitmapFont__
00035
00036 #ifdef __cplusplus
00037
00038
00045 class FTGL_EXPORT FTBitmapFont : public FTFont
00046 {
00047     public:
00053         FTBitmapFont(const char* fontFilePath);
00054
00063         FTBitmapFont(const unsigned char *pBufferBytes,
00064                      size_t bufferSizeInBytes);
00065
00069         ~FTBitmapFont();
00070
00071     protected:
00081         virtual FTGlyph* MakeGlyph(FT_GlyphSlot slot);
00082 };
00083
00084 #define FTGLBitmapFont FTBitmapFont
00085
00086 #endif //__cplusplus
00087
00088 FTGL_BEGIN_C_DECLS
00089
00098 FTGL_EXPORT FTGLfont *ftglCreateBitmapFont(const char *file);
00099
00109 FTGL_EXPORT FTGLfont *ftglCreateBitmapFontFromMem(const unsigned char *bytes,
00110                                                   size_t len);
00111
00112 FTGL_END_C_DECLS
00113
00114 #endif  //  __FTBitmapFont__
00115
```

4.23 FTGLExtrdFont.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTExtrudeFont

  FTExtrudeFont is a specialisation of the FTFont class for handling extruded Polygon fonts.

Macros

- #define FTGLExtrdFont FTExtrudeFont

Functions

- FTGLfont * ftglCreateExtrudeFont (const char *file)

  Create a specialised FTGLfont object for handling extruded polygon fonts.
- FTGLfont * ftglCreateExtrudeFontFromMem (const unsigned char *bytes, size_t len)

  Create a specialised FTGLfont object for handling extruded polygon fonts from a buffer in memory.

4.23.1 Macro Definition Documentation

4.23.1.1 FTGLExtrdFont

```
#define FTGLExtrdFont FTExtrudeFont
```

Definition at line 85 of file FTGLExtrdFont.h.

4.23.2 Function Documentation

4.23.2.1 ftglCreateExtrudeFont()

```
FTGLfont * ftglCreateExtrudeFont (
            const char * file )
```

Create a specialised FTGLfont object for handling extruded polygon fonts.

Parameters

file	The font file name.

Returns

An FTGLfont* object.

See also

FTGLfont

ftglCreatePolygonFont

4.23.2.2 ftglCreateExtrudeFontFromMem()

```
FTGLfont * ftglCreateExtrudeFontFromMem (
            const unsigned char * bytes,
            size_t len )
```

Create a specialised FTGLfont object for handling extruded polygon fonts from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

bytes	the in-memory buffer
len	the length of the buffer in bytes

Returns

An FTGLfont* object.

4.24 FTGLExtrdFont.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTExtrudeFont__
00034 #define __FTExtrudeFont__
00035
00036 #ifdef __cplusplus
```

```
00037
00038
00046 class FTGL_EXPORT FTExtrudeFont : public FTFont
00047 {
00048     public:
00054         FTExtrudeFont(const char* fontFilePath);
00055
00064         FTExtrudeFont(const unsigned char *pBufferBytes,
00065                       size_t bufferSizeInBytes);
00066
00070         ~FTExtrudeFont();
00071
00072     protected:
00082         virtual FTGlyph* MakeGlyph(FT_GlyphSlot slot);
00083 };
00084
00085 #define FTGLExtrdFont FTExtrudeFont
00086
00087 #endif //__cplusplus
00088
00089 FTGL_BEGIN_C_DECLS
00090
00100 FTGL_EXPORT FTGLfont *ftglCreateExtrudeFont(const char *file);
00101
00111 FTGL_EXPORT FTGLfont *ftglCreateExtrudeFontFromMem(const unsigned char *bytes,
00112                                                    size_t len);
00113
00114 FTGL_END_C_DECLS
00115
00116 #endif // __FTExtrudeFont__
00117
```

4.25 FTGLOutlineFont.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTOutlineFont

  FTOutlineFont is a specialisation of the FTFont class for handling Vector Outline fonts.

Macros

- #define FTGLOutlineFont FTOutlineFont

Functions

- FTGLfont * ftglCreateOutlineFont (const char *file)

  Create a specialised FTGLfont object for handling vector outline fonts.
- FTGLfont * ftglCreateOutlineFontFromMem (const unsigned char *bytes, size_t len)

  Create a specialised FTGLfont object for handling vector outline fonts from a buffer in memory.

4.25.1 Macro Definition Documentation

4.25.1.1 FTGLOutlineFont

```
#define FTGLOutlineFont FTOutlineFont
```

Definition at line 84 of file FTGLOutlineFont.h.

4.25.2 Function Documentation

4.25.2.1 ftglCreateOutlineFont()

```
FTGLfont * ftglCreateOutlineFont (
            const char * file )
```

Create a specialised FTGLfont object for handling vector outline fonts.

Parameters

file	The font file name.

Returns

An FTGLfont* object.

See also

FTGLfont

4.25.2.2 ftglCreateOutlineFontFromMem()

```
FTGLfont * ftglCreateOutlineFontFromMem (
            const unsigned char * bytes,
            size_t len )
```

Create a specialised FTGLfont object for handling vector outline fonts from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

bytes	the in-memory buffer
len	the length of the buffer in bytes

Returns

An FTGLfont* object.

4.26 FTGLOutlineFont.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
```

```
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTOutlineFont__
00034 #define __FTOutlineFont__
00035
00036 #ifdef __cplusplus
00037
00038
00045 class FTGL_EXPORT FTOutlineFont : public FTFont
00046 {
00047     public:
00053         FTOutlineFont(const char* fontFilePath);
00054
00063         FTOutlineFont(const unsigned char *pBufferBytes,
00064                       size_t bufferSizeInBytes);
00065
00069         ~FTOutlineFont();
00070
00071     protected:
00081         virtual FTGlyph* MakeGlyph(FT_GlyphSlot slot);
00082 };
00083
00084 #define FTGLOutlineFont FTOutlineFont
00085
00086 #endif //__cplusplus
00087
00088 FTGL_BEGIN_C_DECLS
00089
00098 FTGL_EXPORT FTGLfont *ftglCreateOutlineFont(const char *file);
00099
00109 FTGL_EXPORT FTGLfont *ftglCreateOutlineFontFromMem(const unsigned char *bytes,
00110                                                    size_t len);
00111
00112 FTGL_END_C_DECLS
00113
00114 #endif // __FTOutlineFont__
00115
```

4.27 FTGLPixmapFont.h File Reference

`#include <FTGL/ftgl.h>`

Data Structures

- class FTPixmapFont

  FTPixmapFont is a specialisation of the FTFont class for handling Pixmap (Grey Scale) fonts.

Macros

- #define FTGLPixmapFont FTPixmapFont

Functions

- FTGLfont * ftglCreatePixmapFont (const char *file)

  Create a specialised FTGLfont object for handling pixmap (grey scale) fonts.
- FTGLfont * ftglCreatePixmapFontFromMem (const unsigned char *bytes, size_t len)

  Create a specialised FTGLfont object for handling pixmap (grey scale) fonts from a buffer in memory.

4.27.1 Macro Definition Documentation

4.27.1.1 FTGLPixmapFont

```
#define FTGLPixmapFont FTPixmapFont
```

Definition at line 84 of file FTGLPixmapFont.h.

4.27.2 Function Documentation

4.27.2.1 ftglCreatePixmapFont()

```
FTGLfont * ftglCreatePixmapFont (
            const char * file )
```

Create a specialised FTGLfont object for handling pixmap (grey scale) fonts.

Parameters

file	The font file name.

Returns

An FTGLfont* object.

See also

FTGLfont

4.27.2.2 ftglCreatePixmapFontFromMem()

```
FTGLfont * ftglCreatePixmapFontFromMem (
            const unsigned char * bytes,
            size_t len )
```

Create a specialised FTGLfont object for handling pixmap (grey scale) fonts from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

bytes	the in-memory buffer
len	the length of the buffer in bytes

Returns

An FTGLfont* object.

4.28 FTGLPixmapFont.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTPixmapFont__
00034 #define __FTPixmapFont__
00035
00036 #ifdef __cplusplus
00037
00038
00045 class FTGL_EXPORT FTPixmapFont : public FTFont
00046 {
00047     public:
00053         FTPixmapFont(const char* fontFilePath);
00054
00063         FTPixmapFont(const unsigned char *pBufferBytes,
00064                      size_t bufferSizeInBytes);
00065
00069         ~FTPixmapFont();
00070
00071     protected:
00081         virtual FTGlyph* MakeGlyph(FT_GlyphSlot slot);
00082 };
00083
00084 #define FTGLPixmapFont FTPixmapFont
00085
00086 #endif // __cplusplus
00087
00088 FTGL_BEGIN_C_DECLS
00089
00098 FTGL_EXPORT FTGLfont *ftglCreatePixmapFont(const char *file);
00099
00110 FTGL_EXPORT FTGLfont *ftglCreatePixmapFontFromMem(const unsigned char *bytes,
00111                                                   size_t len);
00112
00113 FTGL_END_C_DECLS
00114
00115 #endif  //  __FTPixmapFont__
00116
```

4.29 FTGLPolygonFont.h File Reference

```
#include <FTGL/ftgl.h>
#include <vector>
```

Data Structures

- class FTPolygonFont

  FTPolygonFont is a specialisation of the FTFont class for handling tesselated Polygon Mesh fonts.

Macros

- #define FTGLPolygonFont FTPolygonFont

Functions

- FTGLfont * ftglCreatePolygonFont (const char *file)

  Create a specialised FTGLfont object for handling tesselated polygon mesh fonts.
- FTGLfont * ftglCreatePolygonFontFromMem (const unsigned char *bytes, size_t len)

  Create a specialised FTGLfont object for handling tesselated polygon mesh fonts from a buffer in memory.

4.29.1 Macro Definition Documentation

4.29.1.1 FTGLPolygonFont

```
#define FTGLPolygonFont FTPolygonFont
```

Definition at line 86 of file FTGLPolygonFont.h.

4.29.2 Function Documentation

4.29.2.1 ftglCreatePolygonFont()

```
FTGLfont * ftglCreatePolygonFont (
            const char * file )
```

Create a specialised FTGLfont object for handling tesselated polygon mesh fonts.

Parameters

file	The font file name.

Returns

An FTGLfont* object.

See also

FTGLfont

4.29.2.2 ftglCreatePolygonFontFromMem()

```
FTGLfont * ftglCreatePolygonFontFromMem (
            const unsigned char * bytes,
            size_t len )
```

Create a specialised FTGLfont object for handling tesselated polygon mesh fonts from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

bytes	the in-memory buffer
len	the length of the buffer in bytes

Returns

An FTGLfont* object.

4.30 FTGLPolygonFont.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
```

```
00031 #endif
00032
00033 #ifndef __FTPolygonFont__
00034 #define __FTPolygonFont__
00035
00036 #ifdef __cplusplus
00037
00038 #include <vector>
00039
00046 class FTGL_EXPORT FTPolygonFont : public FTFont
00047 {
00048     public:
00054         FTPolygonFont(const char* fontFilePath);
00055
00064         FTPolygonFont(const unsigned char *pBufferBytes,
00065                       size_t bufferSizeInBytes);
00066
00070         ~FTPolygonFont();
00071
00072
00073     protected:
00083         virtual FTGlyph* MakeGlyph(FT_GlyphSlot slot);
00084 };
00085
00086 #define FTGLPolygonFont FTPolygonFont
00087
00088 #endif //__cplusplus
00089
00090 FTGL_BEGIN_C_DECLS
00091
00101 FTGL_EXPORT FTGLfont *ftglCreatePolygonFont(const char *file);
00102
00113 FTGL_EXPORT FTGLfont *ftglCreatePolygonFontFromMem(const unsigned char *bytes,
00114                                                    size_t len);
00115
00116 FTGL_END_C_DECLS
00117
00118 #endif  //  __FTPolygonFont__
00119
```

4.31 FTGLTextureFont.h File Reference

`#include <FTGL/ftgl.h>`

Data Structures

- class FTTextureFont

  FTTextureFont is a specialisation of the FTFont class for handling Texture mapped fonts.

Macros

- #define FTGLTextureFont FTTextureFont

Functions

- FTGLfont * ftglCreateTextureFont (const char *file)

  Create a specialised FTGLfont object for handling texture-mapped fonts.
- FTGLfont * ftglCreateTextureFontFromMem (const unsigned char *bytes, size_t len)

  Create a specialised FTGLfont object for handling texture-mapped fonts from a buffer in memory.

4.31.1 Macro Definition Documentation

4.31.1.1 FTGLTextureFont

```
#define FTGLTextureFont FTTextureFont
```

Definition at line 84 of file FTGLTextureFont.h.

4.31.2 Function Documentation

4.31.2.1 ftglCreateTextureFont()

```
FTGLfont * ftglCreateTextureFont (
            const char * file )
```

Create a specialised FTGLfont object for handling texture-mapped fonts.

Parameters

file	The font file name.

Returns

An FTGLfont* object.

See also

FTGLfont

4.31.2.2 ftglCreateTextureFontFromMem()

```
FTGLfont * ftglCreateTextureFontFromMem (
            const unsigned char * bytes,
            size_t len )
```

Create a specialised FTGLfont object for handling texture-mapped fonts from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

bytes	the in-memory buffer
len	the length of the buffer in bytes

Returns

An FTGLfont* object.

4.32 FTGLTextureFont.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTTextureFont__
00034 #define __FTTextureFont__
00035
00036 #ifdef __cplusplus
00037
00038
00045 class  FTGL_EXPORT FTTextureFont : public FTFont
00046 {
00047     public:
00053         FTTextureFont(const char* fontFilePath);
00054
00063         FTTextureFont(const unsigned char *pBufferBytes,
00064                       size_t bufferSizeInBytes);
00065
00069         virtual ~FTTextureFont();
00070
00071     protected:
00081         virtual FTGlyph* MakeGlyph(FT_GlyphSlot slot);
00082 };
00083
00084 #define FTGLTextureFont FTTextureFont
00085
00086 #endif //__cplusplus
00087
00088 FTGL_BEGIN_C_DECLS
00089
00098 FTGL_EXPORT FTGLfont *ftglCreateTextureFont(const char *file);
00099
00110 FTGL_EXPORT FTGLfont *ftglCreateTextureFontFromMem(const unsigned char *bytes,
00111                                                    size_t len);
00112
00113 FTGL_END_C_DECLS
00114
00115 #endif // __FTTextureFont__
00116
```

4.33 FTGLTriangleExtractorFont.h File Reference

```
#include <FTGL/ftgl.h>
#include <vector>
```

Data Structures

- class FTTriangleExtractorFont

  FTPolygonFont is a specialisation of the FTFont class for handling tesselated Polygon Mesh fonts.

Macros

- #define FTGLTriangleExtractorFont FTTriangleExtractorFont

Functions

- FTGLfont * ftglCreateTriangleExtractorFont (const char *file)

  Create a specialised FTGLfont object for handling tesselated polygon mesh fonts.
- FTGLfont * ftglCreateTriangleExtractorFontFromMem (const unsigned char *bytes, size_t len)

  Create a specialised FTGLfont object for handling tesselated polygon mesh fonts from a buffer in memory.

4.33.1 Macro Definition Documentation

4.33.1.1 FTGLTriangleExtractorFont

```
#define FTGLTriangleExtractorFont FTTriangleExtractorFont
```

Definition at line 85 of file FTGLTriangleExtractorFont.h.

4.33.2 Function Documentation

4.33.2.1 ftglCreateTriangleExtractorFont()

```
FTGLfont * ftglCreateTriangleExtractorFont (
            const char * file )
```

Create a specialised FTGLfont object for handling tesselated polygon mesh fonts.

Parameters

file	The font file name.

Returns

An FTGLfont* object.

See also

FTGLfont

4.33.2.2 ftglCreateTriangleExtractorFontFromMem()

```
FTGLfont * ftglCreateTriangleExtractorFontFromMem (
            const unsigned char * bytes,
            size_t len )
```

Create a specialised FTGLfont object for handling tesselated polygon mesh fonts from a buffer in memory.

Sets Error flag. The buffer is owned by the client and is NOT copied by FTGL. The pointer must be valid while using FTGL.

Parameters

bytes	the in-memory buffer
len	the length of the buffer in bytes

Returns

An FTGLfont* object.

4.34 FTGLTriangleExtractorFont.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2011 Richard Ulrich <richi@paraeasy.ch>
00005  *
00006  * Permission is hereby granted, free of charge, to any person obtaining
00007  * a copy of this software and associated documentation files (the
00008  * "Software"), to deal in the Software without restriction, including
00009  * without limitation the rights to use, copy, modify, merge, publish,
00010  * distribute, sublicense, and/or sell copies of the Software, and to
00011  * permit persons to whom the Software is furnished to do so, subject to
00012  * the following conditions:
00013  *
00014  * The above copyright notice and this permission notice shall be
00015  * included in all copies or substantial portions of the Software.
00016  *
00017  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00018  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00019  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00020  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00021  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00022  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00023  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00024  */
00025
00026 #ifndef __ftgl__
00027 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00028 #   include <FTGL/ftgl.h>
00029 #endif
00030
00031 #ifndef __FTTriangleExtractorFont__
00032 #define __FTTriangleExtractorFont__
00033
00034 #ifdef __cplusplus
00035
00036 #include <vector>
00037
00044 class FTGL_EXPORT FTTriangleExtractorFont : public FTFont
00045 {
00046     public:
00053         FTTriangleExtractorFont(const char* fontFilePath, std::vector<float>& triangles);
00054
00064         FTTriangleExtractorFont(const unsigned char *pBufferBytes,
00065                         size_t bufferSizeInBytes, std::vector<float>& triangles);
00066
00070         ~FTTriangleExtractorFont();
```

```
00071
00072     protected:
00082         virtual FTGlyph* MakeGlyph(FT_GlyphSlot slot);
00083 };
00084
00085 #define FTGLTriangleExtractorFont FTTriangleExtractorFont
00086
00087 #endif //__cplusplus
00088
00089 FTGL_BEGIN_C_DECLS
00090
00100 FTGL_EXPORT FTGLfont *ftglCreateTriangleExtractorFont(const char *file);
00101
00112 FTGL_EXPORT FTGLfont *ftglCreateTriangleExtractorFontFromMem(const unsigned char *bytes,
00113                                                              size_t len);
00114
00115 FTGL_END_C_DECLS
00116
00117 #endif  //  __FTTriangleExtractorFont__
00118
```

4.35 FTGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTGlyph

  FTGlyph is the base class for FTGL glyphs.

Typedefs

- typedef struct _FTGLglyph FTGLglyph

Functions

- FTGLglyph * ftglCreateCustomGlyph (FTGLglyph *base, void *data, void(*renderCallback)(FTGLglyph *, void *, FTGL_DOUBLE, FTGL_DOUBLE, int, FTGL_DOUBLE *, FTGL_DOUBLE *), void(*destroy↩ Callback)(FTGLglyph *, void *))

  Create a custom FTGL glyph object.
- void ftglDestroyGlyph (FTGLglyph *glyph)

  Destroy an FTGL glyph object.
- void ftglRenderGlyph (FTGLglyph *glyph, FTGL_DOUBLE penx, FTGL_DOUBLE peny, int renderMode, FTGL_DOUBLE *advancex, FTGL_DOUBLE *advancey)

  Render a glyph at the current pen position and compute the corresponding advance.
- float ftglGetGlyphAdvance (FTGLglyph *glyph)

  Return the advance for a glyph.
- void ftglGetGlyphBBox (FTGLglyph *glyph, float bounds[6])

  Return the bounding box for a glyph.
- FT_Error ftglGetGlyphError (FTGLglyph *glyph)

  Query a glyph for errors.

4.35.1 Typedef Documentation

4.35.1.1 FTGLglyph

```
typedef struct _FTGLglyph FTGLglyph
```

Definition at line 134 of file FTGlyph.h.

4.35.2 Function Documentation

4.35.2.1 ftglCreateCustomGlyph()

```
FTGLglyph * ftglCreateCustomGlyph (
            FTGLglyph * base,
            void * data,
            void(*)(FTGLglyph *, void *, FTGL_DOUBLE, FTGL_DOUBLE, int, FTGL_DOUBLE *, FTGL_DOUBLE
*) renderCallback,
            void(*)(FTGLglyph *, void *) destroyCallback )
```

Create a custom FTGL glyph object.

FIXME: maybe get rid of ¨base¨ and have advanceCallback etc. functions

Parameters

base	The base FTGLglyph∗ to subclass.
data	A pointer to private data that will be passed to callbacks.
renderCallback	A rendering callback function.
destroyCallback	A callback function to be called upon destruction.

Returns

An FTGLglyph∗ object.

4.35.2.2 ftglDestroyGlyph()

```
void ftglDestroyGlyph (
            FTGLglyph * glyph )
```

Destroy an FTGL glyph object.

Parameters

glyph	An FTGLglyph∗ object.

4.35.2.3 ftglGetGlyphAdvance()

```
float ftglGetGlyphAdvance (
            FTGLglyph * glyph )
```

Return the advance for a glyph.

Parameters

glyph	An FTGLglyph* object.

Returns

The advance's X component.

4.35.2.4 ftglGetGlyphBBox()

```
void ftglGetGlyphBBox (
            FTGLglyph * glyph,
            float bounds[6] )
```

Return the bounding box for a glyph.

Parameters

glyph	An FTGLglyph* object.
bounds	An array of 6 float values where the bounding box's lower left near and upper right far 3D coordinates will be stored.

4.35.2.5 ftglGetGlyphError()

```
FT_Error ftglGetGlyphError (
            FTGLglyph * glyph )
```

Query a glyph for errors.

Parameters

glyph	An FTGLglyph* object.

Returns

The current error code.

4.35.2.6 ftglRenderGlyph()

```
void ftglRenderGlyph (
            FTGLglyph * glyph,
```

```
            FTGL_DOUBLE penx,
            FTGL_DOUBLE peny,
            int renderMode,
            FTGL_DOUBLE * advancex,
            FTGL_DOUBLE * advancey )
```

Render a glyph at the current pen position and compute the corresponding advance.

Parameters

glyph	An FTGLglyph* object.
penx	The current pen's X position.
peny	The current pen's Y position.
renderMode	Render mode to display
advancex	A pointer to an FTGL_DOUBLE where to write the advance's X component.
advancey	A pointer to an FTGL_DOUBLE where to write the advance's Y component.

4.36 FTGlyph.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTGlyph__
00034 #define __FTGlyph__
00035
00036 #ifdef __cplusplus
00037
00038 class FTGlyphImpl;
00039
00050 class FTGL_EXPORT FTGlyph
00051 {
00052     protected:
00058         FTGlyph(FT_GlyphSlot glyph);
00059
00060     private:
00067         FTGlyph(FTGlyphImpl *pImpl);
00068
00069         /* Allow our internal subclasses to access the private constructor */
00070         friend class FTBitmapGlyph;
00071         friend class FTBufferGlyph;
```

```
00072         friend class FTExtrudeGlyph;
00073         friend class FTOutlineGlyph;
00074         friend class FTPixmapGlyph;
00075         friend class FTPolygonGlyph;
00076         friend class FTTextureGlyph;
00077         friend class FTTriangleExtractorGlyph;
00078
00079     public:
00083         virtual ~FTGlyph();
00084
00092         virtual const FTPoint& Render(const FTPoint& pen, int renderMode) = 0;
00093
00099         virtual float Advance() const;
00100
00106         virtual const FTBBox& BBox() const;
00107
00113         virtual FT_Error Error() const;
00114
00115     private:
00119         FTGlyphImpl *impl;
00120 };
00121
00122 #endif //__cplusplus
00123
00124 FTGL_BEGIN_C_DECLS
00125
00133 struct _FTGLGlyph;
00134 typedef struct _FTGLglyph FTGLglyph;
00135
00146 FTGL_EXPORT FTGLglyph *ftglCreateCustomGlyph(FTGLglyph *base, void *data,
00147     void (*renderCallback) (FTGLglyph *, void *, FTGL_DOUBLE, FTGL_DOUBLE,
00148                             int, FTGL_DOUBLE *, FTGL_DOUBLE *),
00149     void (*destroyCallback) (FTGLglyph *, void *));
00150
00156 FTGL_EXPORT void ftglDestroyGlyph(FTGLglyph *glyph);
00157
00171 FTGL_EXPORT void ftglRenderGlyph(FTGLglyph *glyph, FTGL_DOUBLE penx,
00172                                  FTGL_DOUBLE peny, int renderMode,
00173                                  FTGL_DOUBLE *advancex, FTGL_DOUBLE *advancey);
00180 FTGL_EXPORT float ftglGetGlyphAdvance(FTGLglyph *glyph);
00181
00189 FTGL_EXPORT void ftglGetGlyphBBox(FTGLglyph *glyph, float bounds[6]);
00190
00197 FTGL_EXPORT FT_Error ftglGetGlyphError(FTGLglyph* glyph);
00198
00199 FTGL_END_C_DECLS
00200
00201 #endif  //  __FTGlyph__
00202
```

4.37 FTLayout.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTLayout

  FTLayout is the interface for layout managers that render text.

Typedefs

- typedef struct _FTGLlayout FTGLlayout

Functions

- void ftglDestroyLayout (FTGLlayout *layout)

  Destroy an FTGL layout object.
- void ftglGetLayoutBBox (FTGLlayout *layout, const char *string, float bounds[6])

  Get the bounding box for a string.
- void ftglRenderLayout (FTGLlayout *layout, const char *string, int mode)

  Render a string of characters.
- FT_Error ftglGetLayoutError (FTGLlayout *layout)

  Query a layout for errors.

4.37.1 Typedef Documentation

4.37.1.1 FTGLlayout

```
typedef struct _FTGLlayout FTGLlayout
```

Definition at line 151 of file FTLayout.h.

4.37.2 Function Documentation

4.37.2.1 ftglDestroyLayout()

```
void ftglDestroyLayout (
            FTGLlayout * layout )
```

Destroy an FTGL layout object.

Parameters

layout	An FTGLlayout* object.

4.37.2.2 ftglGetLayoutBBox()

```
void ftglGetLayoutBBox (
            FTGLlayout * layout,
            const char * string,
            float bounds[6] )
```

Get the bounding box for a string.

Parameters

layout	An FTGLlayout* object.
string	A char buffer
bounds	An array of 6 float values where the bounding box's lower left near and upper right far 3D coordinates will be stored.

4.37.2.3 ftglGetLayoutError()

```
FT_Error ftglGetLayoutError (
            FTGLlayout * layout )
```

Query a layout for errors.

Parameters

layout	An FTGLlayout* object.

Returns

The current error code.

4.37.2.4 ftglRenderLayout()

```
void ftglRenderLayout (
            FTGLlayout * layout,
            const char * string,
            int mode )
```

Render a string of characters.

Parameters

layout	An FTGLlayout* object.
string	Char string to be output.
mode	Render mode to display.

4.38 FTLayout.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
```

```
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTLayout__
00034 #define __FTLayout__
00035
00036 #ifdef __cplusplus
00037
00038
00039 class FTLayoutImpl;
00040
00052 class FTGL_EXPORT FTLayout
00053 {
00054     protected:
00055         FTLayout();
00056
00057     private:
00064         FTLayout(FTLayoutImpl *pImpl);
00065
00066         /* Allow our internal subclasses to access the private constructor */
00067         friend class FTSimpleLayout;
00068
00069     public:
00073         virtual ~FTLayout();
00074
00085         virtual FTBBox BBox(const char* string, const int len = -1,
00086                             FTPoint position = FTPoint()) = 0;
00087
00098         virtual FTBBox BBox(const wchar_t* string, const int len = -1,
00099                             FTPoint position = FTPoint()) = 0;
00100
00111         virtual void Render(const char *string, const int len = -1,
00112                             FTPoint position = FTPoint(),
00113                             int renderMode = FTGL::RENDER_ALL) = 0;
00114
00125         virtual void Render(const wchar_t *string, const int len = -1,
00126                             FTPoint position = FTPoint(),
00127                             int renderMode = FTGL::RENDER_ALL) = 0;
00128
00134         virtual FT_Error Error() const;
00135
00136     private:
00140         FTLayoutImpl *impl;
00141 };
00142
00143 #endif //__cplusplus
00144
00145 FTGL_BEGIN_C_DECLS
00146
00150 struct _FTGLlayout;
00151 typedef struct _FTGLlayout FTGLlayout;
00152
00158 FTGL_EXPORT void ftglDestroyLayout(FTGLlayout* layout);
00159
00168 FTGL_EXPORT void ftglGetLayoutBBox(FTGLlayout *layout, const char* string,
00169                                    float bounds[6]);
00170
00178 FTGL_EXPORT void ftglRenderLayout(FTGLlayout *layout, const char *string,
00179                                   int mode);
00180
00187 FTGL_EXPORT FT_Error ftglGetLayoutError(FTGLlayout* layout);
00188
00189 FTGL_END_C_DECLS
00190
00191 #endif  /* __FTLayout__ */
00192
```

4.39 FTLibrary.h File Reference

```
#include <ft2build.h>
#include <FT_FREETYPE_H>
#include ¨FTGL/ftgl.h¨
#include <atomic>
```

Data Structures

- class FTLibrary

  FTLibrary class is the global accessor for the Freetype library.

4.40 FTLibrary.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  *
00006  * Permission is hereby granted, free of charge, to any person obtaining
00007  * a copy of this software and associated documentation files (the
00008  * "Software"), to deal in the Software without restriction, including
00009  * without limitation the rights to use, copy, modify, merge, publish,
00010  * distribute, sublicense, and/or sell copies of the Software, and to
00011  * permit persons to whom the Software is furnished to do so, subject to
00012  * the following conditions:
00013  *
00014  * The above copyright notice and this permission notice shall be
00015  * included in all copies or substantial portions of the Software.
00016  *
00017  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00018  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00019  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00020  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00021  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00022  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00023  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00024  */
00025
00026 #ifndef     __FTLibrary__
00027 #define     __FTLibrary__
00028
00029 #ifdef __cplusplus
00030
00031 #include <ft2build.h>
00032 #include FT_FREETYPE_H
00033 //#include FT_CACHE_H
00034
00035 #include "FTGL/ftgl.h"
00036 #include <atomic>
00037
00056 class FTLibrary
00057 {
00058     public:
00064         static FTLibrary& Instance();
00065
00071         const FT_Library* GetLibrary() const { return library; }
00072
00078         FT_Error Error() const { return err; }
00079
00085         ~FTLibrary();
00086
00096         void LegacyOpenGLState(bool On);
00097         bool GetLegacyOpenGLStateSet() const { return LegacyOpenGLStateHandling; }
00098
00099     private:
00106         FTLibrary();
00107         FTLibrary(const FT_Library&){}
00108         FTLibrary& operator=(const FT_Library&) { return *this; }
00109
00121         bool Initialise();
00122
00126         FT_Library* library;
00127 //      FTC_Manager* manager;
00128
00132         FT_Error err;
00133
00137         std::atomic <int> LegacyOpenGLStateHandling;
00138 };
00139
00140 #endif //__cplusplus
00141
00142 #endif  //  __FTLibrary__
```

4.41 FTOutlineGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTOutlineGlyph

  FTOutlineGlyph is a specialisation of FTGlyph for creating outlines.

Functions

- FTGLglyph * ftglCreateOutlineGlyph (FT_GlyphSlot glyph, float outset, int useDisplayList)

  Create a specialisation of FTGLglyph for creating outlines.

4.41.1 Function Documentation

4.41.1.1 ftglCreateOutlineGlyph()

```
FTGLglyph * ftglCreateOutlineGlyph (
            FT_GlyphSlot glyph,
            float outset,
            int useDisplayList )
```

Create a specialisation of FTGLglyph for creating outlines.

Parameters

glyph	The Freetype glyph to be processed
outset	outset contour size
useDisplayList	Enable or disable the use of Display Lists for this glyph `true` turns ON display lists. `false` turns OFF display lists.

Returns

An FTGLglyph* object.

4.42 FTOutlineGlyph.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
```

```
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTOutlineGlyph__
00034 #define __FTOutlineGlyph__
00035
00036 #ifdef __cplusplus
00037
00038
00042 class FTGL_EXPORT FTOutlineGlyph : public FTGlyph
00043 {
00044     public:
00056         FTOutlineGlyph(FT_GlyphSlot glyph, float outset, bool useDisplayList);
00057
00061         virtual ~FTOutlineGlyph();
00062
00070         virtual const FTPoint& Render(const FTPoint& pen, int renderMode);
00071 };
00072
00073 #endif //__cplusplus
00074
00075 FTGL_BEGIN_C_DECLS
00076
00088 FTGL_EXPORT FTGLglyph *ftglCreateOutlineGlyph(FT_GlyphSlot glyph, float outset,
00089                                 int useDisplayList);
00090
00091 FTGL_END_C_DECLS
00092
00093 #endif  // __FTOutlineGlyph__
00094
```

4.43 FTPixmapGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTPixmapGlyph

  FTPixmapGlyph is a specialisation of FTGlyph for creating pixmaps.

Functions

- FTGLglyph * ftglCreatePixmapGlyph (FT_GlyphSlot glyph)

  Create a specialisation of FTGLglyph for creating pixmaps.

4.43.1 Function Documentation

4.43.1.1 ftglCreatePixmapGlyph()

```
FTGLglyph * ftglCreatePixmapGlyph (
            FT_GlyphSlot glyph )
```

Create a specialisation of FTGLglyph for creating pixmaps.

Parameters

glyph	The Freetype glyph to be processed

Returns

An FTGLglyph* object.

4.44 FTPixmapGlyph.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTPixmapGlyph__
00034 #define __FTPixmapGlyph__
00035
00036 #ifdef __cplusplus
00037
00038
00042 class  FTGL_EXPORT FTPixmapGlyph : public FTGlyph
00043 {
00044         static FTGlyphImpl *NewImpl(FT_GlyphSlot glyph);
00045
00046     public:
00052         FTPixmapGlyph(FT_GlyphSlot glyph);
00053
00057         virtual ~FTPixmapGlyph();
00058
00066         virtual const FTPoint& Render(const FTPoint& pen, int renderMode);
00067 };
00068
00069 #endif //__cplusplus
00070
00071 FTGL_BEGIN_C_DECLS
00072
00079 FTGL_EXPORT FTGLglyph *ftglCreatePixmapGlyph(FT_GlyphSlot glyph);
00080
00081 FTGL_END_C_DECLS
00082
00083 #endif  //  __FTPixmapGlyph__
00084
```

4.45 FTPoint.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTPoint

  FTPoint class is a basic 3-dimensional point or vector.

4.46 FTPoint.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTPoint__
00034 #define __FTPoint__
00035
00036 #ifdef __cplusplus
00037
00038
00042 class FTGL_EXPORT FTPoint
00043 {
00044     public:
00048         inline FTPoint()
00049         {
00050             values[0] = 0;
00051             values[1] = 0;
00052             values[2] = 0;
00053         }
00054
00062         inline FTPoint(const FTGL_DOUBLE x, const FTGL_DOUBLE y,
00063                        const FTGL_DOUBLE z = 0)
00064         {
00065             values[0] = x;
00066             values[1] = y;
00067             values[2] = z;
00068         }
00069
00075         inline FTPoint(const FT_Vector& ft_vector)
00076         {
00077             values[0] = ft_vector.x;
00078             values[1] = ft_vector.y;
00079             values[2] = 0;
```

```
00080         }
00081
00088         FTPoint Normalise();
00089
00090
00097         inline FTPoint& operator += (const FTPoint& point)
00098         {
00099             values[0] += point.values[0];
00100             values[1] += point.values[1];
00101             values[2] += point.values[2];
00102
00103             return *this;
00104         }
00105
00112         inline FTPoint operator + (const FTPoint& point) const
00113         {
00114             FTPoint temp;
00115             temp.values[0] = values[0] + point.values[0];
00116             temp.values[1] = values[1] + point.values[1];
00117             temp.values[2] = values[2] + point.values[2];
00118
00119             return temp;
00120         }
00121
00128         inline FTPoint& operator -= (const FTPoint& point)
00129         {
00130             values[0] -= point.values[0];
00131             values[1] -= point.values[1];
00132             values[2] -= point.values[2];
00133
00134             return *this;
00135         }
00136
00143         inline FTPoint operator - (const FTPoint& point) const
00144         {
00145             FTPoint temp;
00146             temp.values[0] = values[0] - point.values[0];
00147             temp.values[1] = values[1] - point.values[1];
00148             temp.values[2] = values[2] - point.values[2];
00149
00150             return temp;
00151         }
00152
00159         inline FTPoint operator * (double multiplier) const
00160         {
00161             FTPoint temp;
00162             temp.values[0] = values[0] * multiplier;
00163             temp.values[1] = values[1] * multiplier;
00164             temp.values[2] = values[2] * multiplier;
00165
00166             return temp;
00167         }
00168
00169
00177         inline friend FTPoint operator * (double multiplier, FTPoint& point)
00178         {
00179             return point * multiplier;
00180         }
00181
00182
00190         inline friend double operator * (FTPoint &a, FTPoint& b)
00191         {
00192             return a.values[0] * b.values[0]
00193                  + a.values[1] * b.values[1]
00194                  + a.values[2] * b.values[2];
00195         }
00196
00197
00204         inline FTPoint operator ^ (const FTPoint& point)
00205         {
00206             FTPoint temp;
00207             temp.values[0] = values[1] * point.values[2]
00208                            - values[2] * point.values[1];
00209             temp.values[1] = values[2] * point.values[0]
00210                            - values[0] * point.values[2];
00211             temp.values[2] = values[0] * point.values[1]
00212                            - values[1] * point.values[0];
00213             return temp;
00214         }
00215
00216
00224         friend bool operator == (const FTPoint &a, const FTPoint &b);
00225
00226
00234         friend bool operator != (const FTPoint &a, const FTPoint &b);
00235
00236
```

```
00240         inline operator const FTGL_DOUBLE*() const
00241         {
00242             return values;
00243         }
00244
00245
00249         inline void X(FTGL_DOUBLE x) { values[0] = x; };
00250         inline void Y(FTGL_DOUBLE y) { values[1] = y; };
00251         inline void Z(FTGL_DOUBLE z) { values[2] = z; };
00252
00253
00257         inline FTGL_DOUBLE X() const { return values[0]; };
00258         inline FTGL_DOUBLE Y() const { return values[1]; };
00259         inline FTGL_DOUBLE Z() const { return values[2]; };
00260         inline FTGL_FLOAT Xf() const { return static_cast<FTGL_FLOAT>(values[0]); };
00261         inline FTGL_FLOAT Yf() const { return static_cast<FTGL_FLOAT>(values[1]); };
00262         inline FTGL_FLOAT Zf() const { return static_cast<FTGL_FLOAT>(values[2]); };
00263
00264     private:
00268         FTGL_DOUBLE values[3];
00269 };
00270
00271 #endif //__cplusplus
00272
00273 #endif  //  __FTPoint__
00274
```

4.47 FTPolyGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTPolygonGlyph

  FTPolygonGlyph is a specialisation of FTGlyph for creating tessellated polygon glyphs.

Macros

- #define FTPolyGlyph FTPolygonGlyph

Functions

- FTGLglyph * ftglCreatePolygonGlyph (FT_GlyphSlot glyph, float outset, int useDisplayList)

  Create a specialisation of FTGLglyph for creating tessellated polygon glyphs.

4.47.1 Macro Definition Documentation

4.47.1.1 FTPolyGlyph

```
#define FTPolyGlyph FTPolygonGlyph
```

Definition at line 74 of file FTPolyGlyph.h.

4.47.2 Function Documentation

4.47.2.1 ftglCreatePolygonGlyph()

```
FTGLglyph * ftglCreatePolygonGlyph (
            FT_GlyphSlot glyph,
            float outset,
            int useDisplayList )
```

Create a specialisation of FTGLglyph for creating tessellated polygon glyphs.

Parameters

glyph	The Freetype glyph to be processed
outset	outset contour size
useDisplayList	Enable or disable the use of Display Lists for this glyph `true` turns ON display lists. `false` turns OFF display lists.

Returns

An FTGLglyph* object.

4.48 FTPolyGlyph.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTPolygonGlyph__
00034 #define __FTPolygonGlyph__
00035
00036 #ifdef __cplusplus
00037
00038
00043 class FTGL_EXPORT FTPolygonGlyph : public FTGlyph
00044 {
00045     public:
00057         FTPolygonGlyph(FT_GlyphSlot glyph, float outset, bool useDisplayList);
00058
00062         virtual ~FTPolygonGlyph();
00063
00071         virtual const FTPoint& Render(const FTPoint& pen, int renderMode);
00072 };
00073
00074 #define FTPolyGlyph FTPolygonGlyph
00075
00076 #endif //__cplusplus
00077
00078 FTGL_BEGIN_C_DECLS
00079
00092 FTGL_EXPORT FTGLglyph *ftglCreatePolygonGlyph(FT_GlyphSlot glyph, float outset,
00093                                               int useDisplayList);
00094
00095 FTGL_END_C_DECLS
00096
00097 #endif  //  __FTPolygonGlyph__
00098
```

4.49 FTSimpleLayout.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTSimpleLayout

  FTSimpleLayout is a specialisation of FTLayout for simple text boxes.

Functions

- FTGLlayout * ftglCreateSimpleLayout (void)
- void ftglSetLayoutFont (FTGLlayout *, FTGLfont *)
- FTGLfont * ftglGetLayoutFont (FTGLlayout *)
- void ftglSetLayoutLineLength (FTGLlayout *, const float)
- float ftglGetLayoutLineLength (FTGLlayout *)
- void ftglSetLayoutAlignment (FTGLlayout *, const int)
- int ftglGetLayoutAlignment (FTGLlayout *)
- int ftglGetLayoutAlignement (FTGLlayout *)
- void ftglSetLayoutLineSpacing (FTGLlayout *, const float)
- float ftglGetLayoutLineSpacing (FTGLlayout *)

4.49.1 Function Documentation

4.49.1.1 ftglCreateSimpleLayout()

```
FTGLlayout * ftglCreateSimpleLayout (
            void  )
```

4.49.1.2 ftglGetLayoutAlignement()

```
int ftglGetLayoutAlignement (
            FTGLlayout *  )
```

4.49.1.3 ftglGetLayoutAlignment()

```
int ftglGetLayoutAlignment (
            FTGLlayout *  )
```

4.49.1.4 ftglGetLayoutFont()

```
FTGLfont * ftglGetLayoutFont (
            FTGLlayout *  )
```

4.49.1.5 ftglGetLayoutLineLength()

```
float ftglGetLayoutLineLength (
            FTGLlayout *  )
```

4.49.1.6 ftglGetLayoutLineSpacing()

```
float ftglGetLayoutLineSpacing (
            FTGLlayout *  )
```

4.49.1.7 ftglSetLayoutAlignment()

```
void ftglSetLayoutAlignment (
            FTGLlayout * ,
            const int  )
```

4.49.1.8 ftglSetLayoutFont()

```
void ftglSetLayoutFont (
            FTGLlayout * ,
            FTGLfont *  )
```

4.49.1.9 ftglSetLayoutLineLength()

```
void ftglSetLayoutLineLength (
            FTGLlayout * ,
            const float  )
```

4.49.1.10 ftglSetLayoutLineSpacing()

```
void ftglSetLayoutLineSpacing (
            FTGLlayout * ,
            const float  )
```

4.50 FTSimpleLayout.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
```

```
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTSimpleLayout__
00034 #define __FTSimpleLayout__
00035
00036 #ifdef __cplusplus
00037
00038
00039 class FTFont;
00040
00049 class FTGL_EXPORT FTSimpleLayout : public FTLayout
00050 {
00051     public:
00056         FTSimpleLayout();
00057
00061         ~FTSimpleLayout();
00062
00073         virtual FTBBox BBox(const char* string, const int len = -1,
00074                             FTPoint position = FTPoint());
00075
00086         virtual FTBBox BBox(const wchar_t* string, const int len = -1,
00087                             FTPoint position = FTPoint());
00088
00099         virtual void Render(const char *string, const int len = -1,
00100                             FTPoint position = FTPoint(),
00101                             int renderMode = FTGL::RENDER_ALL);
00102
00113         virtual void Render(const wchar_t *string, const int len = -1,
00114                             FTPoint position = FTPoint(),
00115                             int renderMode = FTGL::RENDER_ALL);
00116
00124         void SetFont(FTFont *fontInit);
00125
00129         FTFont *GetFont();
00130
00136         void SetLineLength(const float LineLength);
00137
00141         float GetLineLength() const;
00142
00149         void SetAlignment(const FTGL::TextAlignment Alignment);
00150
00154         FTGL::TextAlignment GetAlignment() const;
00155
00162         void SetLineSpacing(const float LineSpacing);
00163
00167         float GetLineSpacing() const;
00168 };
00169
00170 #endif //__cplusplus
00171
00172 FTGL_BEGIN_C_DECLS
00173
00174 FTGL_EXPORT FTGLlayout *ftglCreateSimpleLayout(void);
00175
00176 FTGL_EXPORT void ftglSetLayoutFont(FTGLlayout *, FTGLfont*);
00177 FTGL_EXPORT FTGLfont *ftglGetLayoutFont(FTGLlayout *);
00178
00179 FTGL_EXPORT void ftglSetLayoutLineLength(FTGLlayout *, const float);
00180 FTGL_EXPORT float ftglGetLayoutLineLength(FTGLlayout *);
00181
00182 FTGL_EXPORT void ftglSetLayoutAlignment(FTGLlayout *, const int);
00183 FTGL_EXPORT int ftglGetLayoutAlignment(FTGLlayout *);
00184 FTGL_EXPORT int ftglGetLayoutAlignement(FTGLlayout *); // old typo
00185
00186 FTGL_EXPORT void ftglSetLayoutLineSpacing(FTGLlayout *, const float);
00187 FTGL_EXPORT float ftglGetLayoutLineSpacing(FTGLlayout *);
00188
00189 FTGL_END_C_DECLS
00190
00191 #endif  /* __FTSimpleLayout__ */
00192
```

4.51 FTTextureGlyph.h File Reference

```
#include <FTGL/ftgl.h>
```

Data Structures

- class FTTextureGlyph

  FTTextureGlyph is a specialisation of FTGlyph for creating texture glyphs.

Functions

- FTGLglyph * ftglCreateTextureGlyph (FT_GlyphSlot glyph, int id, int xOffset, int yOffset, int width, int height)

  Create a specialisation of FTGLglyph for creating pixmaps.

4.51.1 Function Documentation

4.51.1.1 ftglCreateTextureGlyph()

```
FTGLglyph * ftglCreateTextureGlyph (
            FT_GlyphSlot glyph,
            int id,
            int xOffset,
            int yOffset,
            int width,
            int height )
```

Create a specialisation of FTGLglyph for creating pixmaps.

Parameters

glyph	The Freetype glyph to be processed.
id	The id of the texture that this glyph will be drawn in.
xOffset	The x offset into the parent texture to draw this glyph.
yOffset	The y offset into the parent texture to draw this glyph.
width	The width of the parent texture.
height	The height (number of rows) of the parent texture.

Returns

An FTGLglyph* object.

4.52 FTTextureGlyph.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
```

```
00003  *
00004  * Copyright (c) 2001-2004 Henry Maddocks <ftgl@opengl.geek.nz>
00005  * Copyright (c) 2008 Sam Hocevar <sam@hocevar.net>
00006  * Copyright (c) 2008 Sean Morrison <learner@brlcad.org>
00007  *
00008  * Permission is hereby granted, free of charge, to any person obtaining
00009  * a copy of this software and associated documentation files (the
00010  * "Software"), to deal in the Software without restriction, including
00011  * without limitation the rights to use, copy, modify, merge, publish,
00012  * distribute, sublicense, and/or sell copies of the Software, and to
00013  * permit persons to whom the Software is furnished to do so, subject to
00014  * the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be
00017  * included in all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00020  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00021  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00022  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00023  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00024  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00025  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00026  */
00027
00028 #ifndef __ftgl__
00029 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00030 #   include <FTGL/ftgl.h>
00031 #endif
00032
00033 #ifndef __FTTextureGlyph__
00034 #define __FTTextureGlyph__
00035
00036 #ifdef __cplusplus
00037
00038
00043 class FTGL_EXPORT FTTextureGlyph : public FTGlyph
00044 {
00045     public:
00059         FTTextureGlyph(FT_GlyphSlot glyph, int id, int xOffset, int yOffset,
00060                        int width, int height);
00061
00065         virtual ~FTTextureGlyph();
00066
00074         virtual const FTPoint& Render(const FTPoint& pen, int renderMode);
00075 };
00076
00077 #endif //__cplusplus
00078
00079 FTGL_BEGIN_C_DECLS
00080
00092 FTGL_EXPORT FTGLglyph *ftglCreateTextureGlyph(FT_GlyphSlot glyph, int id,
00093                                               int xOffset, int yOffset,
00094                                               int width, int height);
00095
00096 FTGL_END_C_DECLS
00097
00098 #endif  //  __FTTextureGlyph__
00099
```

4.53 FTTriangleExtractorGlyph.h File Reference

```
#include <FTGL/ftgl.h>
#include <vector>
```

Data Structures

- class FTTriangleExtractorGlyph

  FTPolygonGlyph is a specialisation of FTGlyph for creating tessellated polygon glyphs.

Macros

- #define FTPolyGlyph FTPolygonGlyph

Functions

- FTGLglyph * ftglCreateTriangleExtractorGlyph (FT_GlyphSlot glyph, float outset, int useDisplayList)

  Create a specialisation of FTGLglyph for creating tessellated polygon glyphs.

4.53.1 Macro Definition Documentation

4.53.1.1 FTPolyGlyph

```
#define FTPolyGlyph FTPolygonGlyph
```

Definition at line 73 of file FTTriangleExtractorGlyph.h.

4.53.2 Function Documentation

4.53.2.1 ftglCreateTriangleExtractorGlyph()

```
FTGLglyph * ftglCreateTriangleExtractorGlyph (
            FT_GlyphSlot glyph,
            float outset,
            int useDisplayList )
```

Create a specialisation of FTGLglyph for creating tessellated polygon glyphs.

Parameters

glyph	The Freetype glyph to be processed
outset	outset contour size
useDisplayList	Enable or disable the use of Display Lists for this glyph `true` turns ON display lists. `false` turns OFF display lists.

Returns

An FTGLglyph* object.

4.54 FTTriangleExtractorGlyph.h

Go to the documentation of this file.

```
00001 /*
00002  * FTGL - OpenGL font library
00003  *
00004  * Copyright (c) 2011 Richard Ulrich <richi@paraeasy.ch>
00005  *
00006  * Permission is hereby granted, free of charge, to any person obtaining
00007  * a copy of this software and associated documentation files (the
00008  * "Software"), to deal in the Software without restriction, including
00009  * without limitation the rights to use, copy, modify, merge, publish,
00010  * distribute, sublicense, and/or sell copies of the Software, and to
00011  * permit persons to whom the Software is furnished to do so, subject to
00012  * the following conditions:
00013  *
00014  * The above copyright notice and this permission notice shall be
00015  * included in all copies or substantial portions of the Software.
```

```
00016  *
00017  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00018  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00019  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
00020  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
00021  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
00022  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
00023  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00024  */
00025
00026 #ifndef __ftgl__
00027 #   warning This header is deprecated. Please use <FTGL/ftgl.h> from now.
00028 #   include <FTGL/ftgl.h>
00029 #endif
00030
00031 #ifndef __FTTriangleExtractorGlyph__
00032 #define __FTTriangleExtractorGlyph__
00033
00034 #ifdef __cplusplus
00035
00036 #include <vector>
00037
00042 class FTGL_EXPORT FTTriangleExtractorGlyph : public FTGlyph
00043 {
00044     public:
00056         FTTriangleExtractorGlyph(FT_GlyphSlot glyph, float outset, std::vector<float>& triangles);
00057
00061         virtual ~FTTriangleExtractorGlyph();
00062
00070         virtual const FTPoint& Render(const FTPoint& pen, int renderMode);
00071 };
00072
00073 #define FTPolyGlyph FTPolygonGlyph
00074
00075 #endif //__cplusplus
00076
00077 FTGL_BEGIN_C_DECLS
00078
00091 FTGL_EXPORT FTGLglyph *ftglCreateTriangleExtractorGlyph(FT_GlyphSlot glyph, float outset,
00092                                                         int useDisplayList);
00093
00094 FTGL_END_C_DECLS
00095
00096 #endif  //  __FTTriangleExtractorGlyph__
00097
```

Index